On Robotic Assembly using Contact Force Control and Estimation

Andreas Stolt

LUND
UNIVERSITY

Department of Automatic Control

PhD Thesis
ISRN LUTFD2/TFRT--1109--SE
ISBN 978-91-7623-456-3 (print)
ISBN 978-91-7623-457-0 (web)
ISSN 0280–5316

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

Printed in Sweden by Media-Tryck.
Lund 2015

To Gustav and Viktor

Abstract

Force sensing provides robots the capability to accomplish tasks where physical interaction with the environment is required, such as assembly. Small position uncertainties can then be corrected for by sensing the contact forces. This thesis considers the problem of force-controlled assembly, including how tasks can be specified in a simple and intuitive way and how robust task execution under uncertainties can be accomplished.

A framework for performing robotic assembly is presented. An assembly tasks is composed of a number of skills, where skills both can be force controlled and be carried out using standard position-based control. The skills using force control are specified as sequences of constrained motions, where transitions between the motions are triggered by sensor events. These events can either be simple threshold levels, or be more advanced classifiers based on machine learning. A method for explicitly modeling and resolving uncertainties is presented, as well as a method for adaptation of force control parameters based on identification of a contact model. Specification of sensor-based skills usually requires expert knowledge. To make the specification procedure more easy and intuitive, this thesis presents a method where force-controlled skills can be specified on a high level, and where an executable low-level description is generated. Experimental implementations of multiple assembly scenarios are used to validate the methods and to investigate the potential for force-controlled assembly with industrial robots.

A force sensor may not always be available. The thesis presents two different methods for performing force estimation, based on the measured joint motor angles and the joint motor torques. Friction in the joints is the major disturbance when doing force estimation. A method to increase the accuracy of force estimation using dithering to decrease the effective friction level is proposed. Lead-through programming, to manually guide the robot, is useful for simple and intuitive robot programming. The thesis presents a method for performing such lead-through programming with-

out any force sensor, based on disabling the low-level joint controllers, only feedforwarding the torque to compensate gravity.

Specification and execution of tasks based on external sensing is difficult for non-experts. The methods presented in this thesis all contribute to making it easier and more intuitive to use industrial robots for performing assembly tasks.

Acknowledgments

First of all I thank my supervisors Rolf Johansson and Anders Robertsson, who have always been supporting in my work and have been given me guidance and encouragement. I am further thankful for having been introduced into the field of robotics. Although getting a robot to do what you want is challenging, primarily due to all practical considerations in the lab, it is very rewarding when you succeed and can see the result. Thanks also goes to Klas Nilsson, for always being helpful with everything, from theoretical to practical aspects of all robotics related issues.

Much of my work has been performed in close cooperation with Magnus Linderoth, who has been a very good work partner. His concern for details and theoretical issues has been very rewarding in our cooperation and led to the solution of many problems. I am especially thankful for the proofreading of all my work, this thesis included. Magnus has also made me appreciate hiking in general, and mountain hiking in particular, where he forced me to conquer my fears for mountains during a conference trip to Alaska.

I also thank my colleagues in the ROSETTA project, within which the major part of the research presented in this thesis was performed. I have had many interesting discussions and collaborations, as well as received valuable feedback on my work.

The Department of Automatic Control offers a very nice place to work at, and I am very thankful for being a part of this. The research environment is very stimulating, and the regular 'fika' times encourages discussions about every possible topic. The social events, such as the yearly Christmas and crayfish parties have been very much appreciated. Further, administrative matters always run smoothly, and the computers and other equipment are almost always working as intended thanks to the administrative and technical staff.

I am further thankful to be working together with all colleagues in the Robotics Lab. I am especially honored to be part of "The Guys from Lund", a prestigious title awarded at a summer school in Bavaria 2011,

together with Magnus Linderoth, Olof Sörnmo, Björn Olofsson, and Karl Berntorp. Every now and then, I get reminded of this event when visiting conferences etc. Special thanks also goes to Fredrik Bagge, Martin Karlsson, and Maj Stenmark for good cooperation and for help with this thesis, both regarding proofreading and improving the quality of the graphics.

I also thank Olof Sörnmo, Martin Hast, Josefin Berner, and Meike Stemmann for sharing office with me. You have all contributed to creating a good working environment, where every possible issue can be discussed. The working days have been unpredictable, as you always have to watch your back to avoid practical jokes. For instance, if you leave your computer unlocked, the keyboard layout may be in Arabic when you return, and 2.5 years after my return from parental leave some of my office furniture is still wrapped in magazine paper.

Finally I thank my wife Emma, my sons Gustav and Viktor, my family, and friends. I am especially thankful for all help during the writing of this thesis, with the premature birth of Viktor as an extraordinary circumstance.

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007–2013 – Challenge 2 – Cognitive Systems, Interaction, Robotics – under grant agreement No 230902 - ROSETTA. This document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained herein.

The author is a member of the LCCC Linnaeus Center, supported by the Swedish Research Council, and the eLLIIT Excellence Center at Lund University, supported by the Swedish Government.

Contents

1

Introduction

1.1 Motivation and Background

Traditional industrial robots are most commonly position controlled. Their task is to follow predefined trajectories, where the only sensors used are the motor angle sensors in the joints. Modern robot controllers are very good at performing these tasks, being both fast and achieving good accuracy. Typical robotized applications include welding, painting, packaging, and palletizing. The introduction of robots in industry has relieved human workers from repetitive and/or dangerous tasks, and the robots have become indispensable in some sectors, the automotive industry being one such example.

A classical robot installation usually requires much effort to be spent on structuring the environment around the robot. This includes, e.g., making sure that objects are delivered to or located at known positions, and tooling are tailored for the specific task. The programming is normally time consuming, where much effort is spent on identifying and handling commonly occurring variations. The work spent on setting up a robotic workcell are in spite of these efforts usually economically profitable, due to the robots' ability to be precise and work fast, and the commonly large volume of products to produce.

Some types of tasks are less robotized, and thus still mostly relying on human labor. One such category of tasks are those that require physical interaction between the robot and the environment. These tasks are difficult to handle for a robot using position control. Although the robot is very accurate, a small error in position may lead to a very large force error if the environment is stiff. Assembly is an example of a task that is not that robotized. The reason for this is that several types of uncertainties make it difficult to use the traditional position-controlled approach. Things as part variations, inaccurate gripping, and small tolerances make it hard to achieve the desired accuracy a position-controlled implementation would need. A remedy may be to use different kinds of fixtures and

specialized procedures, e.g., for grasping to make sure that the object ends up at exactly the same position in the gripper each time. Such an implementation, however, is very time consuming, both regarding the design of the workcell and the programming of the robot, and may also put restrictions or induce costs in the actual design of the product. These solutions are also inflexible. If something changes, e.g., if a component in the assembly is replaced by another one produced in a different cheaper material, the properties of the new component may mean that some of the specialized procedures do not work as intended any more. Reusing a position-controlled program is also not especially convenient. Although an assembly task is similar to another, the positions used for the task will most likely all have to be reprogrammed for the second task. The current trend is further going towards more short-series production, and allowing customized products. This means more changes in the tasks, and thus more time for programming the robots will be required, unless new methodologies and techniques are introduced.

A current trend in robotics is that robots are going from being heavy, stiff, very accurate, and stationary, to being mobile and more light-weight. These new robots are further designed to be safe to use in proximity to humans by reducing their mass and power, and making the structure somewhat compliant. These new features open up possibilities for human-robot collaboration, in contrast to the traditional industrial robot that needs to work behind physical or electronic safety fences. Some of these new robots include the PR2 [Willow Garage, 2015], Baxter [Rethink Robotics, 2015], and Nextage [Kawada Industries, 2015]. The ABB YuMi [ABB Robotics, 2015d] is another example of this type of robot, which also is one of the experimental platforms in this thesis. This new type of robot is beneficial to use in assembly operations. The lower inertia combined with the compliant structure will mean that lower forces will be exerted by this kind of robots, making them less prone to damage equipment. The compliance, however, also means that the accuracy will be worse compared to a traditional robot. There will hence be more uncertainty which will have to be handled.

Introducing additional sensing is a way to make robots useful in a wider context, to make them able to function in an unstructured environment, or at least in an environment less structured than today. Sensors are needed to enable the robots to detect events occurring around them. Further, cognition is needed for the robots to be able to interpret the sensor data and to be able to choose appropriate reaction strategies. In other words, the robots need to become more human-like.

For robots to be better suited for accomplishing assembly tasks, force sensing is beneficial. A force sensor can give the robot capability to compensate for insufficient position accuracy, both of the robot and the work-

cell, by sensing the contact forces. The workcell does not even have to be as structured as before, i.e., less work has to be spent on designing fixtures etc. Force sensing makes it also possible to handle other types of uncertainty, e.g., part variations. All magic, however, comes with a price; additional sensing makes the specification of tasks a harder problem, more advanced than to just follow a predefined trajectory. The sensor data must be interpreted and related to the motion of the robot, and in the case of force sensing, force controllers that are tuned for both stability and performance are needed. These things make it really difficult for a non-expert robot user to specify advanced sensor-based implementations of tasks. Easy and intuitive ways to teach robot programs are needed, both for standard position-based programs and sensor-based. Lead-through programming is one such approach, where the user manually guides the robot to either demonstrate the task, or to teach positions for the robot program.

A force sensor does not only contribute to making the task specification difficult, it is commonly also quite expensive. It would be useful if the internal sensing in the robot, usually including joint position sensors and motor torques, could be used for estimating the external forces, and thus make it possible to perform sensorless force control. One potential usage is the above mentioned lead-through programming, which is useful for teaching robot programs. Investing in a force sensor just to be able to perform the teaching of programs, however, might not be economically justified, and a sensorless approach might be a solution.

The research problem considered in this thesis is how to perform robotic assembly. This includes how to specify tasks in a simple way, as well as how to accomplish robust execution. The approach used in the thesis is based on force control, and how force sensing can be used for adaptation to uncertainties and different environments. A force sensor may not always be available, and the thesis also considers the problem of using sensorless force control and estimation as an alternative to a force sensor. Throughout the thesis, experiments are performed to show the usefulness of the contributions.

1.2 Publications

The thesis is based on the following publications.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2011). “Force controlled assembly of emergency stop button”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2011)*. Shanghai, China, pp. 3751–3756.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2012). "Force controlled robotic assembly without a force sensor". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2012)*. St. Paul, MN, USA, pp. 1538–1543.

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

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2013). "Robotic assembly of emergency stop buttons". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2013)*. Video contribution (URL: http://www.youtube.com/watch?v=7JgdbFW5mEg). Tokyo, Japan.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2015). "Detection of contact force transients in robotic assembly". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2015)*. Seattle, WA, USA, pp. 962–968.

The above publications have A. Stolt and M. Linderoth as main contributors and equal contribution is asserted. The work was produced through close cooperation between the main authors, but A. Stolt had a focus toward implementation and experimental evaluation while M. Linderoth had a focus toward the theoretical aspects of the methods. A. Robertsson and R. Johansson assisted with discussion of the ideas and structuring of the manuscripts. The paper "Force controlled robotic assembly without a force sensor" recieved the Best Automation Paper Award at ICRA 2012, and the paper "Detection of contact force transients in robotic assembly" was a finalist for the Best Automation Paper Award at ICRA 2015.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2012). "Adaptation of force control parameters in robotic assembly". In: *Proc. IFAC Symp. Robot Control (SYROCO 2012)*. Dubrovnik, Croatia, pp. 561–566.

Stolt, A., A. Robertsson, and R. Johansson (2015). "Robotic force estimation using dithering to decrease the low velocity friction uncertainties". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2015)*. Seattle, WA, USA, pp. 3896–3902.

The above publications have A. Stolt as the main contributor, and the co-authors assisted with discussing the ideas and structuring of the manuscripts.

Stenmark, M. and A. Stolt (2013). “A system for high-level task specification using complex sensor-based skills”. In: *Robotics: Science and Systems (RSS) 2013 Workshop on Programming with Constraints*. Berlin, Germany.

Stenmark, M., J. Malec, and A. Stolt (2014). “From high-level task descriptions to executable robot code”. In: *IEEE Intelligent Systems ’2014*. Vol. 323. Warsaw, Poland, pp. 189–202.

The above publications have M. Stenmark and A. Stolt as main contributors. M. Stenmark was focused on the high-level programming side, while A. Stolt was focused on the low-level work. J. Malec assisted with discussing the ideas and structuring of the manuscripts.

Stolt, A., F. Bagge Carlson, M. M. Ghazaei Ardakani, I. Lundberg, A. Robertsson, and R. Johansson (2015). “Sensorless friction-compensated passive lead-through programming for industrial robots”. In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2015)*. Hamburg, Germany.

The above publication has A. Stolt as main contributor. M. Ghazaei had the initial idea of the passive lead-through programming, and I. Lundberg discovered the increased sensitivity to external torques by increasing the integral gain. The co-authors assisted with discussing the ideas and structuring of the manuscript.

All of the above publications are available for download from http://www.control.lth.se/Publications.html.

Other Publications

The following publications, where the author also has made contributions in related areas, were decided not to be part of the present thesis.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2012). “Robotic assembly using a singularity-free orientation representation based on quaternions”. In: *Proc. IFAC Symp. Robot Control (SYROCO 2012)*. Dubrovnik, Croatia, pp. 549–554.

The above publication describes how a singularity-free orientation representation can be used within the assembly framework that is presented in this thesis.

Björkelund, A., L. Edström, M. Haage, J. Malec, K. Nilsson, P. Nugues, 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: *Proc. IEEE Int. Symp. Assembly and Manufacturing (ISAM 2011)*. Tampere, Finland, pp. 1–9.

The above publication treats one of the assembly scenarios that is considered in this thesis, and the same assembly framework as is described in this thesis was used for one of the implementations.

Jonsson, M., T. Murray, A. Robertsson, A. Stolt, G. Ossbahr, and K. Nilsson (2010). "Force feedback for assembly of aircraft structures". In: *Proc. SAE Aerospace Manufacturing and Automated Fastening Conference*. Wichita, KS, USA.

Stolt, A., M. Linderoth, A. Robertsson, M. Jonsson, and T. Murray (2011). "Force controlled assembly of flexible aircraft structure". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2011)*. Shanghai, China, pp. 6027–6032.

Jonsson, M., A. Stolt, A. Robertsson, T. Murray, and K. Nilsson (2011). "Force controlled assembly of a compliant rib". In: *SAE2011 Aerotech Congress & Exibition*. Toulouse, France.

Jonsson, M., A. Stolt, A. Robertsson, S. von Gegerfelt, and K. Nilsson (2013). "On force control for assembly and deburring of castings". *Production Engineering* **7**:4, pp. 351–360.

The above publications considers an assembly scenario from the aircraft industry. The same assembly framework that is described in this thesis was used in these publications.

Ceriani, N. M., A. M. Zanchettin, P. Rocco, A. Stolt, and A. Robertsson (2013). "A constraint-based strategy for task-consistent safe human-robot interaction". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2015)*. Tokyo, Japan, pp. 4630–4635.

Ceriani, N., A. Zanchettin, P. Rocco, A. Stolt, and A. Robertsson (2015). "Reactive task adaptation based on hierarchical constraints classification for safe industrial robots". *IEEE/ASME Transactions on Mechatronics* 99. DOI: 10.1109/TMECH.2015.2415462.

The above publications is about safe human-robot interaction in the context of assembly, where the experiments are based upon the assembly framework presented in this thesis.

1.3 Outline and Contributions

The first part of this thesis, Chapters 3–7, presents work on force control. The second part, Chapters 8–10, describes work done in the context of force estimation and sensorless force control. Finally, conclusions are given in Chapter 11.

The chapters have been written so they should be possible to read independently. This comes at the cost of some repetition between the chapters.

Chapter 2 – Hardware and Interfaces

The robots and other hardware used in the experiments in this thesis are presented in this chapter, together with the interface available for control.

Chapter 3 – Robotic Assembly

The framework used for specifying and executing assembly tasks are the topic of this chapter. Two different assembly use cases illustrate the use of the framework. The implementations of the use cases are compared to the state of the art in assembly, namely the human assembly worker.

Chapter 4 – Uncertainty Estimation in Robotic Assembly

This chapter presents how uncertainties can be modeled and resolved within the assembly framework. Two different uncertainties from an assembly task are used to illustrate the methodology, and it is shown in experiments how the uncertainites can be resolved.

Chapter 5 – Detection of Contact Force Transients

A force controlled assembly task is usually implemented as a sequence of simple motions, where the transitions between these motions are made when new contact situations are detected. These new contact situations results in transients in the measured force/torque data. In this chapter, a systematic procedure for training machine learning based classifiers for detecting these transients is presented. In an example assembly task, it is shown how the method can be used to decrease the cycle time.

Chapter 6 – Generation of Sensor-Based Robot Programs from High-Level Task Specifications

When a human instructs another human how an assembly task should be performed, almost exclusively a high-level description is given. The topic of this chapter is a system that makes it possible to do this also for a robot. The high-level specification is used to generate executable code for the robot. Experiments are used to verify the concept in a realistic assembly setting.

Chapter 7 – Adaptation of Force-Control Parameters

Tuning of force controllers are often tedious work, and it will make it much simpler for for non-expert users if an adaptive functionality is available. In this chapter, such a method for self-tuning of force controllers used for robots is described. The method is also integrated into the assembly framework and experimentally evaluated.

Chapter 8 – Robotic Force Estimation without Force Sensor

Two new methods for estimation of external forces acting on the end-effector of a robot are presented in this chapter. The first method is based on the joint position-control errors, and the second one on the motor torques together with modeling of the low-velocity friction disturbances. Both methods are experimentally evaluated in numerous assembly tasks and compared to each other.

Chapter 9 – Dithering to Improve the Accuracy of Force Estimation

A method to increase the accuracy of force estimation by using dithering to decrease the low-velocity friction disturbances are presented in this chapter. The method is experimentally evaluated and applied in a lead-through programming scenario.

Chapter 10 – Sensorless Friction-Compensated Passive Lead-Through Programming

A method for lead-through programming without the use of a force sensor is the topic of this chapter. The method works by disabling the low-level servo controllers and only feedforward the torques to balance gravity and compensate for friction. As no position- or force-feedback loops are active, the method becomes stable for any environment, no matter how stiff it is. Further, the sensitivity to external forces when the joints are not moving is shown to be improved by enabling the low-level servo controllers with increased integral gain. The lead-through programming method is implemented and experimentally evaluated on two different industrial robots.

2

Hardware and Interfaces

All experiments presented in this thesis have been made in the Robotics Lab of the departments of Automatic Control and Computer Science at Lund University. This chapter gives a brief overview of the hardware and interfaces used.

2.1 Robots

Three different robots have been used in this thesis. The first one was the ABB IRB140 robot [ABB Robotics, 2015b] (see Fig. 2.1), which is a common industrial robot with 6 degrees of freedom. It has a payload of 6 kg, a reach of 810 mm, and a position repeatability of ±0.03 mm. The second robot was the ABB IRB120 [ABB Robotics, 2015a] (see Fig. 2.2), which

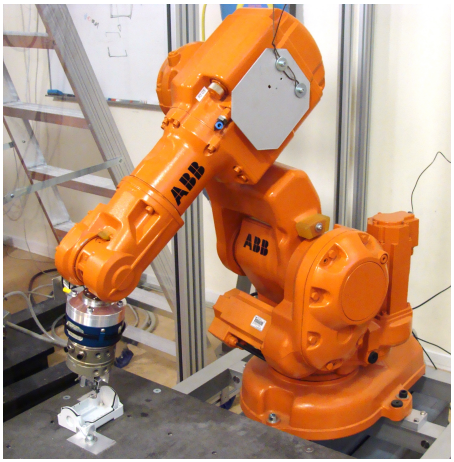


Figure 2.1 The ABB IRB140 robot used in the experiments in this thesis. The robot is equipped with a wrist-mounted JR3 force/torque sensor.

Figure 2.2 The ABB IRB120 robot used in the experiments in this thesis.

is a small industrial robot with 6 degrees of freedom. It has a payload of 3 kg, a reach of 580 mm, and a position repeatability of ±0.01 mm.

The third robot used was ABB YuMi [Kock et al., 2011; ABB Robotics, 2015d] (previously known as FRIDA) (see Fig. 2.3), a dual arm robot designed for performing assembly. Each of the two arms is redundant with 7 degrees of freedom. The robot is weak and lightweight, and much effort has been spent on making it safe to use next to humans [Matthias et al., 2011]. All sharp edges have been covered with soft padding, and together with the low mass and the power and speed limitations the robot is designed not to be able to hurt a human.

2.2 Interface to the robots

The robot systems considered in this thesis were controlled by the ABB IRC5 control system. It had a control structure such that each joint was individually controlled, with a main computer that calculated references for each of the joints. A cascaded control structure was used for each joint, see Fig. 2.4 for a joint model and a block diagram of the control structure. The outer position loop had proportional feedback, while the inner velocity loop had both proportional and integral feedback. The controller parameters were the three gains K_p, K_v, and K_i. There was further a control loop for the electrical current in the motor. The current control loop ensured that the motor actuated the desired torque.

The research interface available to the robots [Blomdell et al., 2005;

Figure 2.3 The ABB YuMi robot used in experiments in this thesis. The robot is in this photo equipped with wrist-mounted ATI Mini40 force/torque sensors.

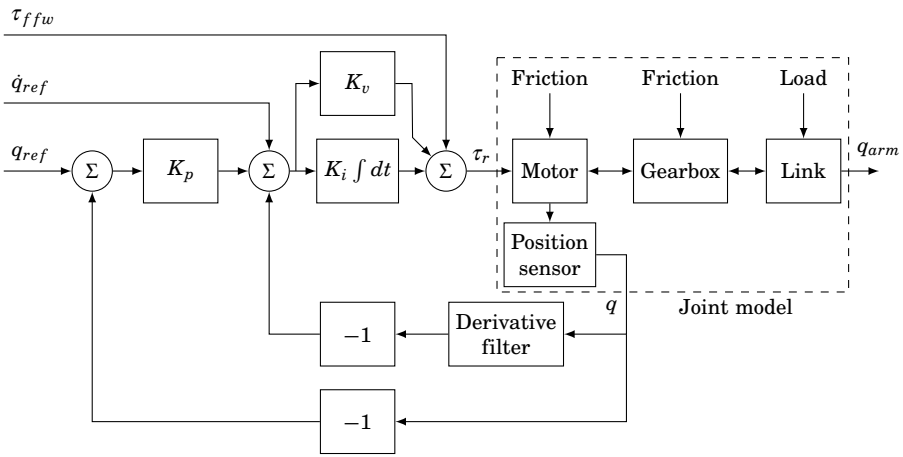


Figure 2.4 Block diagram of the low-level joint controller running at 2 kHz. The joint position q refers to the motor position, while q_{arm} refers to the arm side position. In the joint model, there is a dynamic relation between the link and the motor via the gearbox.

Blomdell et al., 2010] made it possible to alter the signals sent from the main computer, i.e., one could send position and velocity references and a torque feedforward signal. Whereas it was possible to modify the control gains (K_p, K_v, K_i), it was not possible to change the controller structure. Available measurements included joint positions, and velocities as numerically differentiated and filtered positions. The torque reference sent to the motors was also an available signal, which will be close to the actual torque exerted by the motors as the current loop was tightly controlled. The low-level control loops ran with a sampling frequency of 2 kHz, while the research interface for setting the references and reading measurements ran at 250 Hz.

The external controller using the interface was executed on an external PC running with Linux and Xenomai [Xenomai, 2015] for real-time performance. The communication with the robot controller was made with the LabComm protocol [LabComm, 2015], which allows the specification of data types that should be sent over a socket. The communication overhead has been kept to a minimum and the protocol is thus appropriate for sending data in real time.

2.3 Force sensing

The IRB140 robot was equipped with a wrist-mounted six degrees-of-freedom JR3 100M40A force/torque sensor [JR3, 2015]. The workcell with the YuMi robot was equipped with one or two six degrees-of-freedom ATI Mini40 force/torque sensors [ATI, 2015], either one sensor placed on the table, or wrist-mounted sensors for each of the arms.

3

Robotic Assembly

3.1 Introduction

There exist a few different strategies for performing robotic assembly, where different amounts of sensor information are used. One way is to use pure position control of the robot. This approach relies on that the accuracy of the robot, of the parts involved, and of the workcell are sufficiently good. What is meant with sufficiently good varies, but for small-parts assembly tasks it is usually in the sub-millimeter scale, and to achieve this accuracy task specific fixtures and toolings are commonly needed. Further, one has to be certain that nothing unexpected will happen during the assembly operation, as this is hard to discover without an external sensor. It is possible to handle some degree of part variation by using a compliant tool.

A second strategy is to use binary information from sensors. This means that the assembly is divided into several steps, in which the information from the sensors is used to trigger transitions between the steps. This strategy can be used when there are a few uncertainties, e.g., some variations in the parts. One example can be to use a sequence of search movements in order to find a certain feature of an object, and once this feature is found, it is possible to use pure position control to finish the assembly operation.

Yet another alternative is to use sensors for continuous feedback control. This strategy makes it possible to cope with large uncertainties, but it is also the strategy that is the most difficult to program for a robot operator. One example of this strategy is force controlled assembly, where force sensing can be used to identify contact situations, keep contacts and find new contacts during an assembly operation. If contacts only are detected in a binary fashion, as described in the previous paragraph, there is a risk of losing contact or getting very large contact forces during sliding motions. Hence, continuous sensing can make assembly possible when more uncertainties are involved, and also reduce the risk of damaging

equipment. This strategy also makes it possible to reuse programs, as the same program can be used in similar tasks where the sensing can handle and compensate for the differences.

The strategies described in the previous paragraphs can handle different amounts of uncertainties and the type of effort one has to spend to get them running is different. In the case of pure position control one has to assume that the position accuracy is very good, the workcell is well structured with all parts where they are supposed to be, and that everything will go as planned. These requirements can be relaxed when binary sensor information is used, but then one has to take care of the sensor signals in an appropriate way instead. Using continuous sensing demands even more sensor processing, and requires feedback control strategies which may be hard to tune. The two last strategies also require a sensor, which may be expensive. The increased robustness to uncertainties and the possibility for reusability, however, are incitements to use the strategy based on continuous sensing.

This chapter considers the problem of accomplishing assembly tasks in a robust way with industrial robots using force control. A framework for specifying and performing force-controlled assembly tasks is presented. The state of the art for performing assembly tasks is the human assembly worker, and the performance of the robot is therefore evaluated in comparison to the human.

Traditional position-controlled robotic tasks are specified as trajectories for the robot to follow. When external sensors are introduced, this way of specifying tasks is not very good as it is difficult to relate the sensor measurements to the trajectories. An early framework for specifying force-controlled tasks is via hybrid position/force control [Raibert and Craig, 1981], where different control modes are used in different Cartesian directions using selection matrices. An alternative to the hybrid control approach is the parallel force/position controller [Chiaverini and Sciavicco, 1993], where both a force controller and position controller are simultaneously active in each Cartesian direction, but where the force controller is designed to dominate over the position controller. Other frameworks for specifying the end-effector motion are, for instance, the operational space formulation [Khatib, 1987], the use of the compliance frame [Mason, 1981], or the task frame formalism [Bruyninckx and De Schutter, 1996]. All these frameworks specify different controllers for each direction in a certain coordinate frame. The task specification framework used in this thesis is based on the constraint based task specification framework, iTaSC (instantaneous Task Specification using Constraints) [De Schutter et al., 2007], where control modes, or constraints, can be specified in arbitrary directions in task space by the use of so called feature frames. The iTaSC framework makes it possible to incorporate multiple exter-

nal sensors and also take geometric uncertainties into account. The force controllers used in this thesis are impedance controllers [Hogan, 1985], which aim at controlling a dynamic relationship between the robot and the environment.

A survey of the requirements for autonomous robotic assembly is given in [Bruyninckx et al., 2001], where the needed components are reviewed and directions for future research are outlined. The authors focus on explicitly modeling the possible contact formations and how the system should be able to use this information. For non-convex objects however, the number of possible contact formations become huge, which becomes difficult to handle. In this thesis, assembly is accomplished by not explicitly modeling the contact formations. A previous example of robotic assembly can, e.g., be found in [Arai et al., 2006], where optimization of force control parameters with respect to cycle time was made in assembly of a clutch. Another example is presented in [Jörg et al., 2000], which describes an assembly scenario where sensor fusion of vision and force sensing was used to insert cylinders into a rotating engine. Yet another example from the automotive industry is given in [Gravel et al., 2008], which describes powertrain assembly. A peg-in-hole assembly implemented in a dual robot setting is presented in [Caccavale et al., 1998]. One of the arms was performing the insertion of the peg in a position-based manner, while the other arm was holding the hollow part. Experiments show that much better performance was achieved when the arm holding the hollow part was force-controlled to be compliant, as compared to when it was mechanically breaked. The experiments show that force control is beneficial for assembly. An example of assembly from the construction industry where position control was used is given in [Gambao et al., 2000]. An application of force control in robotics other than assembly is [T. Olsson et al., 2010], where force control was used to avoid sliding movements when drilling.

3.2 Task specification and control framework

A hierarchical structure was used for specifying the assembly tasks. A *task* was divided into a number of *skills*, e.g., procedures for picking objects or specific assembly operations. A skill was composed of a number of *motions*, which could be a movement from one position to another, or a search motion in a specific direction. This section presents how force-controlled motions are specified and how they are composed into skills.

Specification of force-controlled motions

The force-controlled motions were specified using the iTaSC-framework [De Schutter et al., 2007; Smits, 2010]. A detailed illustration of how the

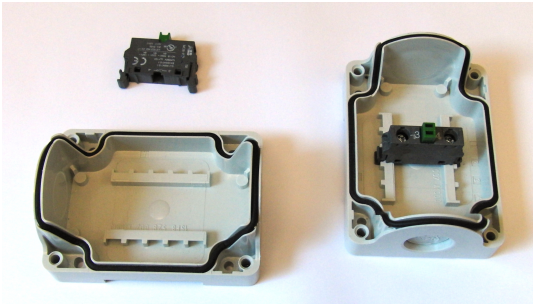


Figure 3.1 The parts involved in the snapfit assembly task are shown to the left, and the final assembled product to the right.

framework can be applied in the context of assembly is given in this thesis, and a flexible implementation that is able to handle general tasks is presented. An example of another implementation of iTaSC is based on OROCOS [*OROCOS* 2015]. The implementation in this thesis uses the original iTaSC definitions and control scheme, but is extended with more controller types, and support for modeling of known time-varying transformations are added.

The iTaSC framework specifies the relative motion of objects by imposing constraints, such as position or force constraints. To be able to specify these constraints in an easy way, kinematic chains are introduced. They contain object and feature frames that are used to simplify the task specification. The modeling procedure is illustrated on the snapfit assembly task, where the involved parts and the final assembled product are displayed in Fig. 3.1. The dark gray electrical switch should be snapped into place in the light gray bottom box. The tolerances are very small, which makes it very difficult to implement this scenario using position control.

The object frames should be attached to the objects that are part of the task. In the snapfit scenario, the relevant objects are the bottom box, the switch, and the robot. The first object frame, $o1$, is attached to a corner of the bottom box, as illustrated in Fig. 3.2. The second object frame, $o2$, is chosen to coincide with the flange[1] frame of the robot.

The feature frames should be used to make the task specification as simple as possible. They should therefore be attached to specific features on the objects that are relevant for the task. In the snapfit scenario, important features are the slot that the switch should be mounted in, and the ends of the switch that should make contact with the slot. The first feature frame, $f1$, is therefore attached to the slot that the switch

[1] The connector plate at the end of the robot arm, where the tool is mounted.

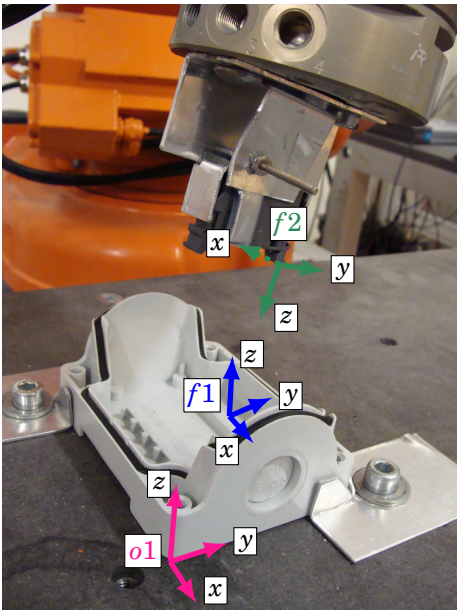


Figure 3.2 Illustration of the coordinate frames in the snapfit scenario. Object frame $o2$ is placed at the flange of the robot, not displayed in this photo.

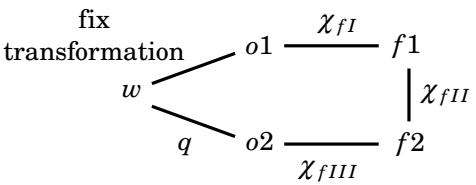


Figure 3.3 Schematic illustration of the kinematic chain used in the snapfit assembly task.

should be placed in. The second feature frame, $f2$, is attached to the end of the switch that should be put into the slot where $f1$ is placed.

Further, one needs to specify a world coordinate frame, w. In the snapfit scenario this is chosen to coincide with the base frame of the robot. The object frames can now be given relations to w, a fix transformation for $o1$, and a transformation depending on the robot joint coordinates, q, for $o2$. In the case that the bottom box would be traveling on a conveyor belt, the fix transformation can be replaced by a time-varying transformation parameterized by χ_k, the known coordinates. The transformation

between $o1$ and $o2$, via $f1$ and $f2$, should have 6 degrees of freedom that parametrize the transformation. These degrees of freedom are called the feature coordinates, χ_f; they are further divided into χ_{fI}, χ_{fII}, and χ_{fIII}, denoting the coordinates between $o1$ and $f1$, $f1$ and $f2$, and $f2$ and $o2$, respectively. A schematic illustration of the kinematic chain is given in Fig. 3.3. In the snapfit scenario, $f1$ is fix relative to $o1$, and $f2$ is fix relative to $o2$, i.e., all degrees of freedom are in the transformation between $f1$ and $f2$. The feature coordinates chosen are first three translations along the coordinate axes of $f1$, then three Euler XYZ angles to describe the reorientation from $f1$ to $f2$, i.e, they are given as

$$\chi_{fI} = (-) \quad , \quad \chi_{fII} = (x, y, z, \varphi, \theta, \psi) \quad , \quad \chi_{fIII} = (-) \tag{3.1}$$

Geometric uncertainties can be modeled by introducing uncertainty frames, which represent the modeled position of the frame, e.g., uncertainty frame $o1'$ is the modeled position of the actual frame $o1$. The degrees of freedom in the transformations between the uncertainty frames and the true frames are called the uncertainty coordinates, χ_u. They represent the pose uncertainty in the frame, see further details about uncertainty management in Chap. 4.

The variables to be constrained are chosen by specifying outputs y. In general, each output can be a function of the feature and the robot joint coordinates, but if the kinematic chains have been chosen properly the outputs will in most cases directly correspond to some of the feature coordinates. Multiple kinematic chains may be used to choose appropriate outputs for the task. The outputs are in general defined by

$$y = f(q, \chi_f) \tag{3.2}$$

In the snapfit scenario, the outputs are chosen to be all of the feature coordinates, i.e, they are chosen according to

$$y = \chi_f \tag{3.3}$$

The kinematic chains should be closed, which implies a relation between the robot joint coordinates (q), the feature coordinates (χ_f), the uncertainty coordinates (χ_u), and the known coordinates (χ_k). This relation can be expressed as

$$l(q, \chi_f, \chi_u, \chi_k) = 0 \tag{3.4}$$

Such a function can be expressed by the following position-loop constraint, i.e., a product of homogeneous transformation matrices

$$\begin{aligned} &T_w^{o1'}(q, \chi_k) T_{o1'}^{o1}(\chi_u) T_{o1}^{f1'}(\chi_{fI}) T_{f1'}^{f1}(\chi_u) T_{f1}^{f2}(\chi_{fII}) \ldots \\ &T_{f2}^{f2'}(\chi_u) T_{f2'}^{o2}(\chi_{fIII}) T_{o2}^{o2'}(\chi_u) T_{o2'}^{w}(q, \chi_k) = I_{4\times4} \end{aligned} \tag{3.5}$$

where $I_{4\times4}$ is the 4×4 identity matrix. Note that both $T_w^{o1'}$ and $T_{o2'}^w$ may be functions of q and χ_k. In a dual arm scenario, both of the transformations will be dependent of q, but in a single arm scenario, only one of $T_w^{o1'}$ and $T_{o2'}^w$ will be a function of q. The known coordinates, χ_k, can both be constant, i.e., correspond to a fix transformation, or be time-varying, e.g., modeling a conveyor belt.

Control

The control in this thesis has been performed both on the velocity level and on the acceleration level. The control signal should be references for the low-level servo controllers, either $\dot{q}_{ref}$ or $\ddot{q}_{ref}$, i.e., joint velocity reference or joint acceleration reference. In the sequel, the robot is assumed to track the given references and the subscript ref is dropped.

Velocity level On the velocity level, a control law for the robot joint velocities, $\dot{q}$, has to be derived. The time derivative of (3.2) is

$$\dot{y} = C_q\dot{q} + C_f\dot{\chi}_f \tag{3.6}$$

where $C_q = \partial f/\partial q$ and $C_f = \partial f/\partial\chi_f$ are the Jacobians of f with respect to q and χ_f, respectively. Further, the loop constraints (3.4) have the time derivative

$$J_q\dot{q} + J_f\dot{\chi}_f + J_u\dot{\chi}_u + J_k\dot{\chi}_k = 0 \tag{3.7}$$

where $J_q = \partial l/\partial q$, $J_f = \partial l/\partial\chi_f$, $J_u = \partial l/\partial\chi_u$, and $J_k = \partial l/\partial\chi_k$ are the Jacobians of l with respect to the different coordinates. The feature coordinate time derivatives $\dot{\chi}_f$ can be solved from (3.7), according to

$$\dot{\chi}_f = -J_f^{-1}\left(J_q\dot{q} + J_u\dot{\chi}_u + J_k\dot{\chi}_k\right) \tag{3.8}$$

To be able to do this it is required that J_f is invertible, and this will be the case if the feature coordinates parametrize all 6 degrees of freedom in task space. Equation (3.8) can now be substituted into (3.6), giving

$$A\dot{q} = \dot{y} + B_u\dot{\chi}_u + B_k\dot{\chi}_k \tag{3.9}$$

where $A = C_q - C_fJ_f^{-1}J_q$, $B_u = C_fJ_f^{-1}J_u$, and $B_k = C_fJ_f^{-1}J_k$.

All constraints should be expressed at the velocity level, and this means specifying $\dot{y} = \dot{y}_d^0$, where $\dot{y}_d^0$ is the desired velocity of the outputs. To be able to use more than velocity constraints, $\dot{y}_d^0$ is chosen according to

$$\dot{y}_d^0 = \dot{y}_d + C \tag{3.10}$$

where $\dot{y}_d$ is a feedforward velocity term and C is a feedback controller that might use additional sensing, such as a force sensor.

If the robot is assumed to be an ideal velocity controlled system, the control signal that has to be calculated is $\dot{q}$. This can be made according to

$$\dot{q} = A^{-1}\left(\dot{y}_d^0 + B_u\dot{\hat{\chi}}_u + B_k\dot{\chi}_k\right) \tag{3.11}$$

where $\dot{\hat{\chi}}_u$ denotes the estimate of $\dot{\chi}_u$. Note that A will have to be invertible for this calculation to be possible, which will be the case, e.g., if the task is completely specified (constraints in all 6 degrees of freedom) and the robot has 6 degrees of freedom and is not in a singular configuration. The calculated value of $\dot{q}$ is integrated to get a position reference, and both the position and velocity are sent as control signals to the robot.

Acceleration level When the control is performed on the acceleration level, a control law for the joint accelerations, $\ddot{q}$, has to be derived. This can be performed by using the same approach as in the velocity level control derivation, but taking an extra derivative with respect to time. Taking one more time derivative of (3.6) gives

$$\ddot{y} = C_q\ddot{q} + \dot{C}_q\dot{q} + C_f\ddot{\chi}_f + \dot{C}\dot{\chi}_f \tag{3.12}$$

where $\dot{C}_q = dC_q/dt$ and $\dot{C}_f = dC_f/dt$. Also taking the time derivative of (3.7) gives

$$J_q\ddot{q} + \dot{J}_q\dot{q} + J_f\ddot{\chi}_f + \dot{J}_f\dot{\chi}_f + J_u\ddot{\chi}_u + \dot{J}_u\dot{\chi}_u + J_k\ddot{\chi}_k + \dot{J}_k\dot{\chi}_k = 0 \tag{3.13}$$

The feature coordinate accelerations, $\ddot{\chi}_f$, can now be solved for

$$\ddot{\chi}_f = -J_f^{-1}\left(J_q\ddot{q} + \dot{J}_q\dot{q} + \dot{J}_f\dot{\chi}_f + J_u\ddot{\chi}_u + \dot{J}_u\dot{\chi}_u + J_k\ddot{\chi}_k + \dot{J}_k\dot{\chi}_k\right) \tag{3.14}$$

As in the velocity control case, cf. (3.8), J_f is required to be invertible, which is the case if the feature coordinates for each kinematic chain parametrize all six degrees of freedom in task space. By substituting (3.14) into (3.12), the following equation is achieved

$$A_a\ddot{q} = \ddot{y} + B_{a,q}\dot{q} + B_{a,f}\dot{\chi}_f + B_{a,u1}\ddot{\chi}_u + B_{a,u2}\dot{\chi}_u + B_{a,k1}\ddot{\chi}_k + B_{a,k2}\dot{\chi}_k \tag{3.15}$$

where

$$\begin{array}{lll} A_a = C_q - C_fJ_f^{-1}J_q & B_{a,q} = C_fJ_f^{-1}\dot{J}_q - C_q & B_{a,f} = C_fJ_f^{-1}\dot{J}_f - \dot{C}_f \\ B_{a,u1} = C_fJ_f^{-1}J_u & B_{a,u2} = C_fJ_f^{-1}\dot{J}_u & \\ B_{a,k1} = C_fJ_f^{-1}J_k & B_{a,k2} = C_fJ_f^{-1}\dot{J}_k & \end{array}$$

All constraints should now be expressed on the acceleration level, i.e., specifying $\ddot{y} = \ddot{y}_d^0$. In the same manner as with the velocity level control, $\ddot{y}_d^0$ is chosen according to

$$\ddot{y}_d^0 = \ddot{y}_d + C_a \tag{3.16}$$

where $\ddot{y}_d$ is a feedforward acceleration term and C_a is a feedback controller on the acceleration level, which might use external sensing.

If the robot would have been an acceleration controlled system, the control signal to be sent to it would be (derived from (3.15))

$$\ddot{q} = A_a^{-1}\left(\ddot{y}_d^0 + B_{a,q}\dot{q} + B_{a,f}\dot{\chi}_f + B_{a,u1}\ddot{\hat{\chi}}_u + B_{a,u2}\dot{\hat{\chi}}_u + B_{a,k1}\ddot{\hat{\chi}}_k + B_{a,k2}\dot{\chi}_k\right) \tag{3.17}$$

where $\ddot{\hat{\chi}}_u$ and $\dot{\hat{\chi}}_u$ denotes the estimates of $\ddot{\chi}_u$ and $\dot{\chi}_u$, respectively.

The robots used in the experiments in this thesis are, however, position controlled with possibility of giving a velocity reference as well. The control signal must therefore be the joint positions, q, and the joint velocities, $\dot{q}$, which can be calculated by integrating (3.17) twice

$$\dot{q} = \int \ddot{q}dt \quad , \quad q = \int \dot{q}dt \tag{3.18}$$

A discretization of (3.18) can, e.g., be given by Tustin's method [Åström and Wittenmark, 1996]

$$\begin{aligned} \dot{q}(kh+h) &= \dot{q}(kh) + \tfrac{h}{2}\left(\ddot{q}(kh+h) + \ddot{q}(kh)\right) \\ q(kh+h) &= q(kh) + \tfrac{h}{2}\left(\dot{q}(kh+h) + \dot{q}(kh)\right) \end{aligned} \tag{3.19}$$

where h is the sampling period and k the sampling index.

Redundancy The iTaSC framework is suitable to handle both over- and under-constrained tasks, as well as manipulators with redundant degrees of freedom. The motion specification is calculated by solving for the robot joint velocities, $\dot{q}$, in (3.9), or the joint accelerations, $\ddot{q}$, in (3.15). When the task is redundant, or over-constrained, the matrix A (or A_a) will not be square, and hence a pseudoinverse must be used. In case of a redundant task at the velocity level, the weighted pseudoinverse $A^\dagger$ in (3.20) can be used. The interpretation is that the optimization problem (3.21) is solved, where M is a positive definite weighting matrix. For the acceleration level, the same pseudoinverse can be used, but where A_a replaces A in (3.20), and the corresponding optimization problem will have a quadratic objective function in $\ddot{q}$ with the constraint being (3.15).

$$A^\dagger = M^{-1}A^T\left(AM^{-1}A^T\right)^{-1} \tag{3.20}$$

$$\begin{aligned} &\underset{\text{over } \dot{q}}{\text{minimize}} && \dot{q}^T M \dot{q} \\ &\text{subject to} && A\dot{q} = \dot{y}_d^0 + B\dot{\hat{\chi}}_u \end{aligned} \tag{3.21}$$

Feedback controllers on task level

As mentioned in the previous section, all constraints have to be given at the velocity level (or the acceleration level). Three different kinds of constraints have been used throughout this thesis, namely position, velocity,

and force constraints. Each such constraint was handled by a feedback controller, that outputs an appropriate velocity (acceleration) for the corresponding output. Scalar controllers were used for each component of the output vector y.

Position controller on the velocity level The position controller used was a proportional controller on the velocity level, that for each position controlled output y^i was given as

$$\dot{y}^0_{d,i} = K(y^i_{ref} - y^i) \tag{3.22}$$

where K is the proportional gain and y^i_{ref} is the position reference. The output from the controller was limited to avoid too large velocities when new position references (possibly far away from the current position) were given. The rate of change of the controller was also limited, i.e., limiting the acceleration. No feedforward term was used.

Position controller on the acceleration level A state feedback controller was used, with a double integrator as process model, such that

$$\ddot{x} = u \tag{3.23}$$

where u is the control signal (the acceleration) and x is the position coordinate to control. A state space realization of (3.23) is

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \\ x &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{aligned} \tag{3.24}$$

and a zero-order-hold sampled realization is

$$\begin{aligned} \begin{bmatrix} x_1(kh+h) \\ x_2(kh+h) \end{bmatrix} &= \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(kh) \\ x_2(kh) \end{bmatrix} + \begin{bmatrix} h^2/2 \\ h \end{bmatrix} u(kh) \\ x &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(kh) \\ x_2(kh) \end{bmatrix} \end{aligned} \tag{3.25}$$

where h is the sampling period, and k denotes the sampling index.

The control signal $u(kh)$ is given as

$$u(kh) = L \begin{bmatrix} x_1(kh) - x_{1,ref}(kh) \\ x_2(kh) - x_{2,ref}(kh) \end{bmatrix} + u_{ffw}(kh) \tag{3.26}$$

and the applied control signal, i.e., as defined in Eq. (3.16), is

$$\dot{y}^0_{d,i}(kh) = u(kh) \tag{3.27}$$

where x in (3.23) is assumed to represent output y^i.

The feedforward part, u_{ffw}, was given from a reference generator that calculates the time-optimal trajectory to the position reference, given constraints on velocity and acceleration, see App. A. The feedback part was chosen as a linear quadratic (LQ) controller [Åström and Wittenmark, 1996]. The state reference values, $x_{1,ref}$ and $x_{2,ref}$, were given from the reference generator, and the values of x_1, the value of the controlled output, and x_2, the time derivative of the output, were both available (see the section on model update and estimation on page 36). The feedforward part handles the desired trajectory, and the feedback part takes care of errors, mostly originating from the sampled integration of the acceleration, i.e., the errors associated with performing (3.19).

Velocity controller on the velocity level The velocity controller used was based on pure feedforward, according to

$$\dot{y}^0_{d,i} = \dot{y}^i_{ref} \tag{3.28}$$

where $\dot{y}^i_{ref}$ is the velocity reference. The low-level joint control loops were assumed to track the desired velocity, such that no feedback was needed. To handle large reference changes, the rate of change of the velocity reference $\dot{y}^i_{ref}$ was limited.

Velocity controller on the acceleration level A proportional controller was used, that for each velocity controlled output y^i was given as

$$\dot{y}^0_{d,i} = K\left(\dot{y}^i_{ref} - \dot{y}^i\right) \tag{3.29}$$

where K is the proportional gain and $\dot{y}^i_{ref}$ is the reference, i.e., the desired velocity. The acceleration was limited by saturating $\dot{y}^0_{d,i}$.

Force controller Impedance controllers [Hogan, 1985] were used to handle force constraints. This controller gives the desired acceleration for the output y^i, according to

$$\dot{y}^0_{d,i} = \frac{1}{M}\left(F^i - F^i_{ref} - D\dot{y}^0_{d,i}\right) \tag{3.30}$$

where F^i denotes the force in the direction of y^i, and F^i_{ref} the force reference value. The parameter M is the virtual mass and D the virtual damping of the impedance that the controller acts like. The output acceleration, $\dot{y}^0_{d,i}$, was limited, and also the output velocity. On the acceleration level, the velocity was limited by modifying the acceleration such that the velocity in the next sample would not exceed the maximum velocity. On the velocity level, the limited acceleration was integrated and used as controller output, which also was saturated to limit the velocity.

Controller switching When a switch of controllers for the same output is made, e.g., when a velocity constraint is changed to a force constraint, the output velocity is made continuous by adjusting the initial state of the new controller. This ensures a smooth transfer, as the velocity becomes continuous.

Model update and estimation

The state of the system, i.e., the values of the feature coordinates χ_f and estimates of the uncertainty coordinates χ_u, are calculated in each sample. The position loop constraints (3.5) are used to make sure that the values of χ_f are consistent with the robot joint coordinates, q, which are given as measurements from the robot, and the known coordinates, χ_k. The uncertainty coordinates are first assumed to be constant when the value of χ_f is calculated, and then updated, more about this in Chap. 4.

As it is assumed that the only unknown variable of (3.5) is χ_f, the left hand side can be written as $T(\chi_f)$ and the goal is to achieve

$$T(\chi_f) = I_{4\times4} \tag{3.31}$$

This equation holds if χ_f is known, when this is not the case, the identity matrix will be replaced by

$$T_{err} = \begin{bmatrix} R_{err} & t_{err} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{3.32}$$

Here R_{err} represents the orientation error and t_{err} the translation error. A linear approximation of the error is given by

$$J_f \Delta\chi_f = \begin{bmatrix} t_{err} \\ a_{err} \end{bmatrix} \tag{3.33}$$

where $\Delta\chi_f$ is the error in the feature coordinates, and a_{err} is an axis/angle representation of R_{err}. As J_f is invertible, this relation can be used to calculate a new estimate of χ_f, according to

$$\chi_f^{i+1} = \chi_f^i - \Delta\chi_f = \chi_f^i - J_f^{-1}(\chi_f^i) \begin{bmatrix} t_{err}^i \\ a_{err}^i \end{bmatrix} \tag{3.34}$$

where i denotes the iteration index. One such iteration would be sufficient if the original system was linear. This is not the case and therefore some iterations of this procedure are needed. As the feature coordinates are calculated continuously in each sample, the initial value is the value calculated in the previous sample, and therefore only one or a few iterations are needed for convergence.

The values of the outputs y are straightforwardly calculated using (3.2). The time derivatives of the outputs, $\dot{y}$, can be calculated from (3.9), where the joint velocities, $\dot{q}$, is a measurement from the robot. The forces acting in the feature coordinate directions, F_f, are calculated as

$$F_f = J_f^T F \tag{3.35}$$

where F is the force/torque applied at the end of the kinematic chain as measured by a force sensor. The force applied in the direction of the outputs, F_y, is then calculated as

$$F_y = C_f^{-T} F_f \tag{3.36}$$

where C_f^{-T} must be replaced with a pseudoinverse if the task is not completely specified, i.e., if C_f^T is not invertible, e.g., if the task is underspecificied such that fewer than six outputs are specified.

Skill specification

A skill is composed of a number of motions, each accomplishing a part of an assembly task. The sequencing is modeled by a finite state machine [Gill, 1962]. In the snapfit assembly scenario, uncertainties in part locations and gripping made it impossible to use pure position control to accomplish the assembly task. The strategy was therefore to use a sequence of search motions, where each of these motions were designed to resolve some of the uncertainty. Each state in the state machine contains a set of kinematic chains and constraints specifying the motion. For instance, a linear search motion in the z-direction is described by position constraints for all coordinates except for the z-coordinate, which instead has a velocity constraint. State transitions are based on sensor measurements, e.g., a detected contact force or that a certain position has been reached.

The state machine used for the snapfit skill is displayed in Fig. 3.4, and snapshots from the sequence showing the search directions during an execution of the assembly task are displayed in Fig. 3.5. For the snapfit scenario it was assumed that the position and orientation of the of the bottom box were not known sufficiently well to go straight into the slot. The area in front of the slot, however, was larger and thus possible to hit. Initial contact was therefore established with the bottom of the box in this area (state number two), and the slot was found by two successive search motions (while contact was kept with the bottom). Next, state number five, the switch was rotated around the contact point in the slot, such that it also made contact with the other end of the switch. The initial pose of the switch was chosen such that it was known in which direction

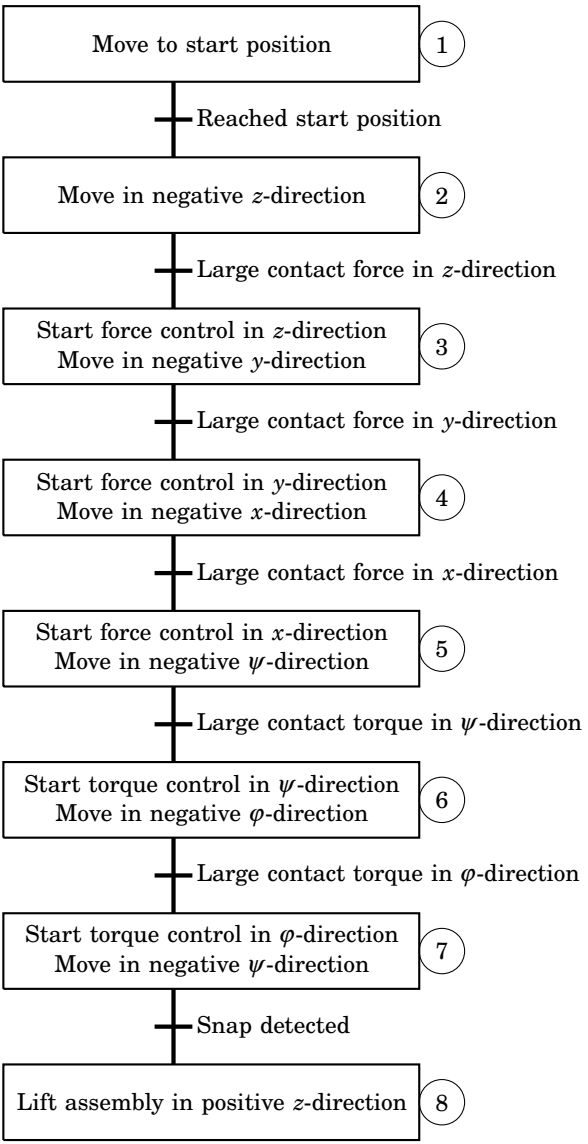


Figure 3.4 The state machine used for the snapfit assembly scenario.

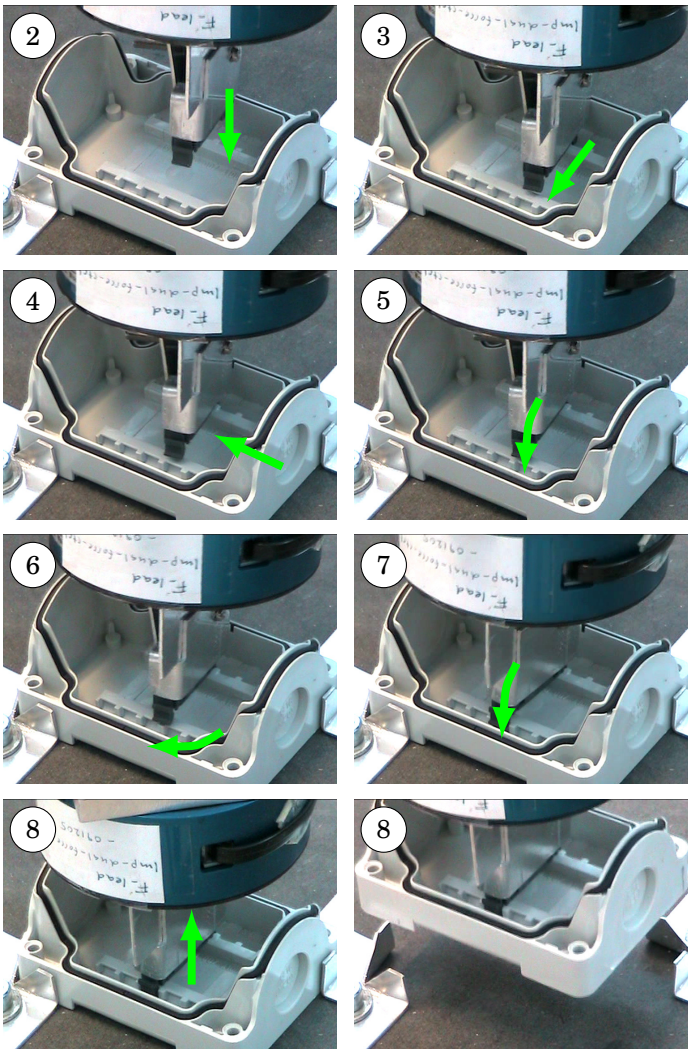


Figure 3.5 Snapshots from an execution of the snapfit assembly scenario. The arrows show the search direction in each state, with state numbers defined in Fig. 3.4.

to turn the switch to find the other side of the slot, and a rotation was performed in this direction until the switch slid down into the slot (a torque was applied on the switch during the rotation). The remaining step was to push the switch completely into the slot.

Software implementation

The assembly framework was implemented using Mathworks' Matlab/Simulink environment [Mathworks, 2015a]. Executable code was generated via the Simulink Coder [Mathworks, 2015b], and the compiled program was run as an external controller using the research interface presented in Chap. 2.

Force-controlled assembly skills were coordinated with finite state machines. Both statecharts using Mathworks' Stateflow [Mathworks, 2015c] and sequential function charts using JGrafchart [JGrafchart, 2015] were used for the implementation. The state machine outputs kinematic chains to use in each state, encoded as lists of simple transformations. Further, the types of constraints to use are communicated, i.e, which type of controller to use. Finally, various different types of parameters are sent, e.g., controller parameters and reference values. The inputs to the state machine are measurements, i.e., values of the outputs, and forces and velocities in the output directions.

The framework was such that it could be integrated with a standard ABB RAPID program [ABB Robotics, 2012]. In this way, position controlled skills such as initial positioning and picking of parts from well-defined trays could be performed with the specialized position control architecture in the ABB controller. When external sensing was needed, the execution could be handed over to the external controller, and the execution could be handed back to the ABB controller once the task using external sensing was finished.

In the implementation, a RAPID program was used for coordinating the execution of a task. A number of skills were used, executed both in sequence and in parallel if possible. Some of the skills were position-controlled RAPID procedures, and some were force-controlled skills, which were executed by handing over the execution to the external controller.

3.3 Emergency stop button use case

An assembly graph for the assembly task is displayed in Fig. 3.6. The assembly can be divided into three main parts.

The red button assembly The parts for this scenario are displayed in the upper right part of Fig. 3.6. First the red button should be inserted

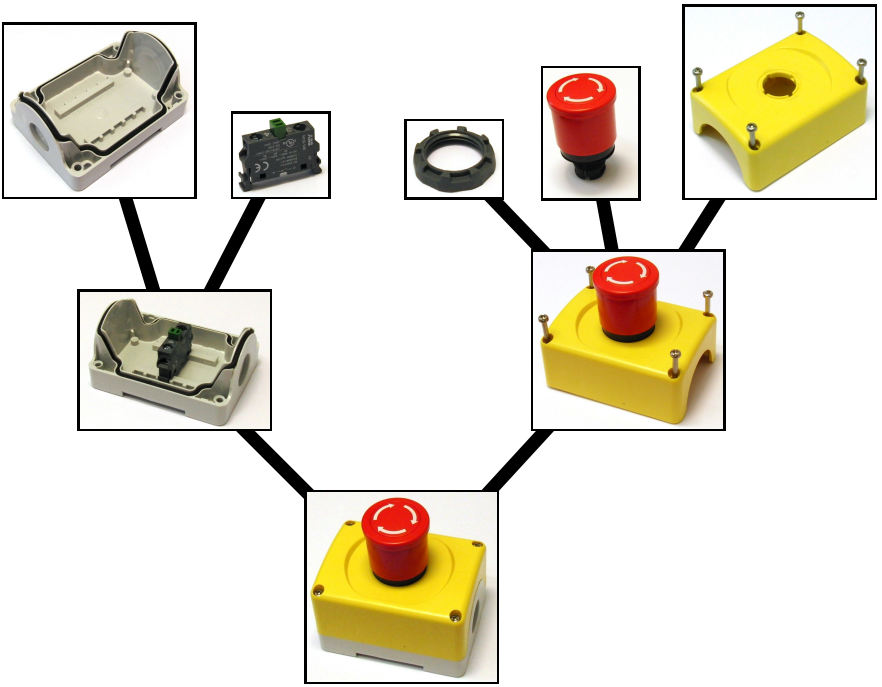


Figure 3.6 Assembly graph for the emergency stop button assembly scenario.

into the hole on top of the yellow case, i.e., a peg-in-hole assembly. Once inserted, the nut should be screwed on the button to attach it to the yellow case.

The snapfit assembly The upper left part of Fig. 3.6 shows the parts for the snapfit assembly scenario. The dark gray electrical switch should be placed in one of five available slots in the light gray bottom box. Each slot is slightly larger than the switch, but flexibility in the switch makes it possible to snap it into the correct position.

The complete assembly With the two sub-assemblies performed, i.e., the snapfit and the red button assemblies, the last operation is to place the yellow case with the button on top of the bottom box with the switch. Finally, the screws should be screwed to finish the assembly, this part of the assembly is, however, not considered in this thesis, as this would require some specialized tooling.

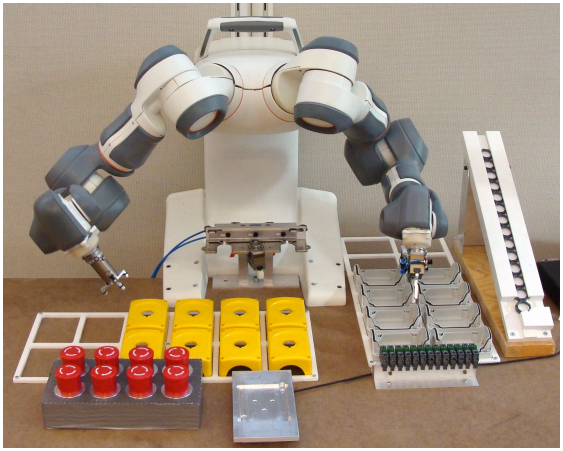


Figure 3.7 The workcell for the emergency stop button use case with the YuMi robot.

Workcell

The assembly scenario was implemented with different robots with different setups. The main structure of the workcell was, however, similar in all cases. A fixture was used to make it possible to do some of the assembly operations using one robot arm. The workcell used for the implementation with the YuMi robot is displayed in Fig. 3.7. The fixture can be seen in the bottom center of the photo. Three different grippers were used in the implementation. A force/torque sensor was placed beneath the fixture, such that force control could be performed when in contact with the fixture. For other operations not performed in contact with the fixture where force control was needed, such as screwing the nut, force estimation as described in Chap. 8 was used.

The snapfit assembly and parts of the red button assembly were also performed using the IRB140. In this setup, a wrist-mounted force/torque sensor was used to perform force control.

Assembly sequence

To complete the assembly, a number of different operations had to be performed. Each of these skills are presented in more detail on the following pages.

- Pick yellow case
- Put yellow case on fixture
- Pick red button

- Insert red button into yellow case
- Pick nut
- Lift red button and yellow case and turn them around
- Screw nut
- Align yellow case against fixture
- Check if yellow case should be turned 180 degrees
- Put yellow case in intermediate storage
- Change tool
- Pick gray box
- Put gray box on fixture
- Pick switch
- Do snapfit
- Put switch and gray box on table
- Pick yellow case from intermediate storage
- Put yellow case on top of gray box and slide them to the side
- Change tool

The left arm of YuMi had to change tool to be able to complete the assembly task. The first tool was used for the red button assembly, and the second tool for the snapfit assembly. To reduce the number of tool changes, a number of red button assemblies were performed and placed in an intermediate storage before the tool was changed.

Position-controlled skills

Some of the skills were implemented as procedures in RAPID, based on pure position control. These skills included all picking skills, which were possible as all parts were placed in well-defined trays. The tool exchange skill was another RAPID based skill, that was performed with a fixed tool stand. Finally, also the last parts of the sequence, including moving the switch and the gray box from the fixture to the table, and putting the yellow case and the button on top of the gray box to finish the assembly, were implemented as RAPID skills.

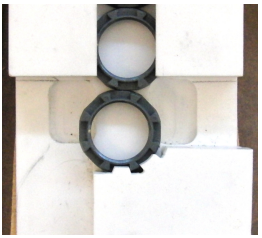


Figure 3.8 A nut in the correct pick position in the slider.

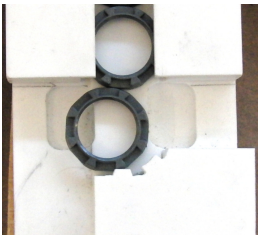


Figure 3.9 A nut that has ended up too far to the left in the slider.

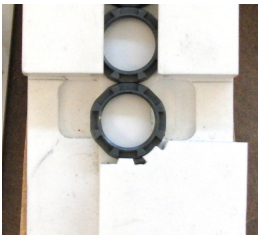


Figure 3.10 A nut that has ended up too far to the right in the slider.

Robust picking of nut The nuts were picked from a slider, which can be seen in the right part of Fig. 3.7. The bottom part of the slider was designed such that the nut to pick should end up in a pre-determined position with a known orientation. When a nut had been picked and the rest of the nuts slid down, however, it was quite common that the next nut to pick ended up in another position than what was expected. Examples of different nut positions are displayed in Figs. 3.8–3.10. As no external feedback signals were available, a robust picking skill was implemented to prevent these situations from causing errors.

The nut could be forced into the pre-determined pick position by pushing on it from the side. As it could end up both to the left and to the right of the intended pick position (Figs. 3.9 and 3.10), the nut had to be pushed from both sides. The pushing procedure was performed before every nut was picked, and the resulting failure rate was very low.

Turning of red button and yellow case This skill was used to lift the yellow case and the red button from the fixture on the force sensor to a position that was suitable for screwing the nut. It was possible to do the turning using only one arm, but the nearby tray with red buttons and kinematic limitations of the robot arm made it very difficult. Instead, the turning was performed using both of YuMi's arms. The arm holding the button first performed a rotation around the edge of the yellow case, such that it was possible for the other arm to help in the lifting from the other side, see photo in Fig. 3.11.

The lifting was performed as a synchronized motion, i.e., such that one of the arms was controlled to keep a constant position and orientation with respect to the other arm. The yellow case and the button were turned such that the button pointed upwards (see Fig. 3.12), as this was a favorable position for the screwing of the nut.

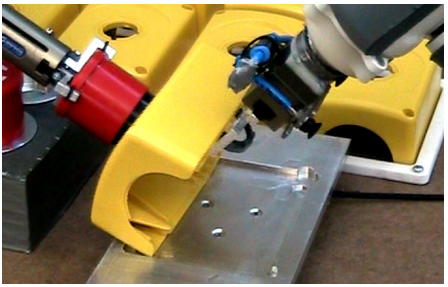


Figure 3.11 Turning of the red button and the yellow case.

Force-controlled skills

Screwing This skill was to fasten the button to the yellow case by screwing the nut. The right arm of the robot held the button with the case on top, and the left arm held the nut and performed the screwing. The arm holding the button was used only for holding the button and the case still. Whereas screwing also with this arm might have increased the assembly speed, uncertainties in the exact position of the button in the gripper made the screwing very unreliable then. The initial configuration before the screwing started is displayed in Fig. 3.12, and the frames used for modeling the skill are illustrated in Figs. 3.12–3.13.

Two kinematic chains were used for the nut screw scenario. The first one (chain a) was used for specifying the relative motion between the arms, and the second (chain b) for specifying the motion of one of the arms with respect to the world coordinate frame, in this case the right arm.

The first chain is illustrated in more detail in Fig. 3.13 (here the superscript a has been dropped). The world coordinate frame, w, was placed at the base of the robot and the chain began with the left YuMi arm. Object frame $o1$ was placed on the flange of the robot arm, and feature frame $f1$ in the middle of the gripped nut. Feature frame $f2$ was placed at the bottom of the yellow case, centered in the button. The second object frame, $o2$, was placed at the flange of the right robot arm. The chain was closed by going back to w through the right arm.

The feature coordinates used were all gathered between $f1$ and $f2$, first three translations along the coordinate axes of $f1$ and then three Euler angles to describe the reorientation. The translations described the relative distance between $f1$ and $f2$, e.g., a positive z-translation resulted in the arms moving apart (at least for the configuration shown in Fig. 3.13).

The second kinematic chain, b, began once again in the robot base.

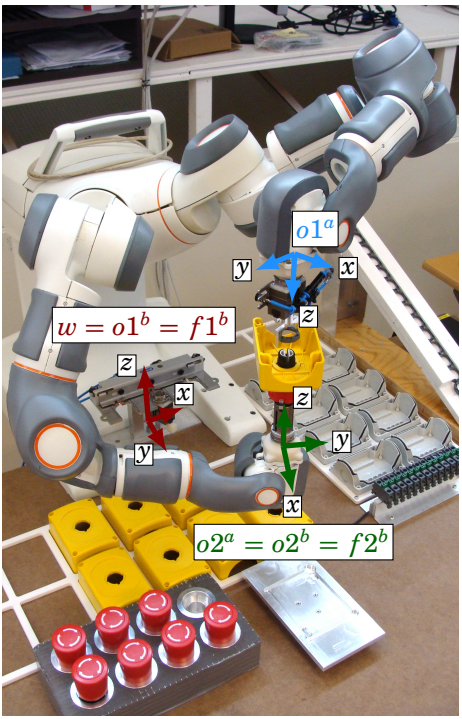


Figure 3.12 Illustration of the two kinematic chains used in the nut screw assembly scenario.

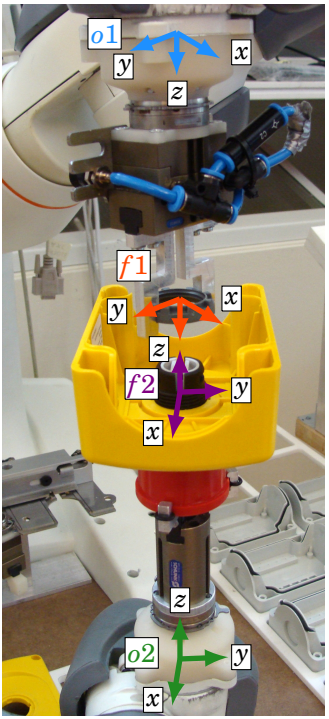


Figure 3.13 A detailed view of the kinematic chain a in the nut screw assembly scenario.

This chain was used to specify the motion of one of the arms, and as no relevant features were present for this task, both $o1^b$ and $f1^b$ were chosen to coincide with the world coordinate frame w. Feature frame $f2^b$ and object frame $o2^b$ were further chosen to coincide with the flange of the right robot arm. The chain was closed by going back to w through the robot arm. The feature coordinates used were three translations and three Euler angles, all gathered between $f1^b$ and $f2^b$.

No force sensors were available on the arms, and the assembly operation therefore had to rely on the estimated forces using the methods described in Chap. 8.

The nominal assembly strategy was as follows

1. Initial positioning
2. Put nut on the button by a search in $f1$ z-direction
3. Re-grip nut

4. Screw until nut is tightened

5. Release nut and move away

Because of various types of uncertainties, this assembly strategy would most likely fail. To make the execution more robust, both proactive and reactive strategies were implemented. A sketch of the actual state machine coordinating the assembly skill is displayed in Fig. 3.14.

To start with, the red button with the yellow case on it was placed in a vertical position. The second arm, with the nut in the gripper, approached the first arm and tried to put the nut on the button (as in Fig. 3.12). This was made as a search in the negative $f1^a$ z-direction. The search was ended when a contact force was detected, and if the distance between the arms was small enough, the nut was put on the button, and this scenario corresponded to the nominal strategy. On the other hand, if the distance between the arms was not small, the hole was missed. The recovery strategy was to move the robot back, slightly modify the position, and try again. This operation continued until the nut was mounted on the button.

The gripper for the nut could grasp the nut in two ways. The first grip was used for putting the nut on the button, and the second for screwing. The nut therefore had to be released and re-gripped, and this might cause a movement of the nut. To be robust against this type of error, a proactive strategy was applied, namely to push on the nut to make sure it was all the way down on the button. The next step was to go down to the nut, grip it, and start screwing. The robot could screw one revolution, and then had to release the nut and go back to the start position again, before the process was repeated.

It was known that it took 2.5 to 3.5 revolutions to tighten the nut, depending on where on thread the screwing was started. Detecting that the nut was tightened could not be done through the estimated torque around the screwing axis, due to too large disturbances. What usually happened was that the grip of the nut was lost when it became tightened, and then the screwing arm pushed on the arm holding the button, resulting in a large side force that could be detected. If this happened during the third or the fourth revolution of screwing, it was assumed that the reason was that the nut was tightened. If no large side forces had occurred after four revolutions, the button had most likely slid in the gripper, which meant that it had been tightened.

During screwing, there was a risk that the nut was gripped in a non-centric way, and this might also cause large side forces although the nut was not tightened. If such a force was detected and less than three revolutions had been screwed, the screwing was stopped. The nut was then

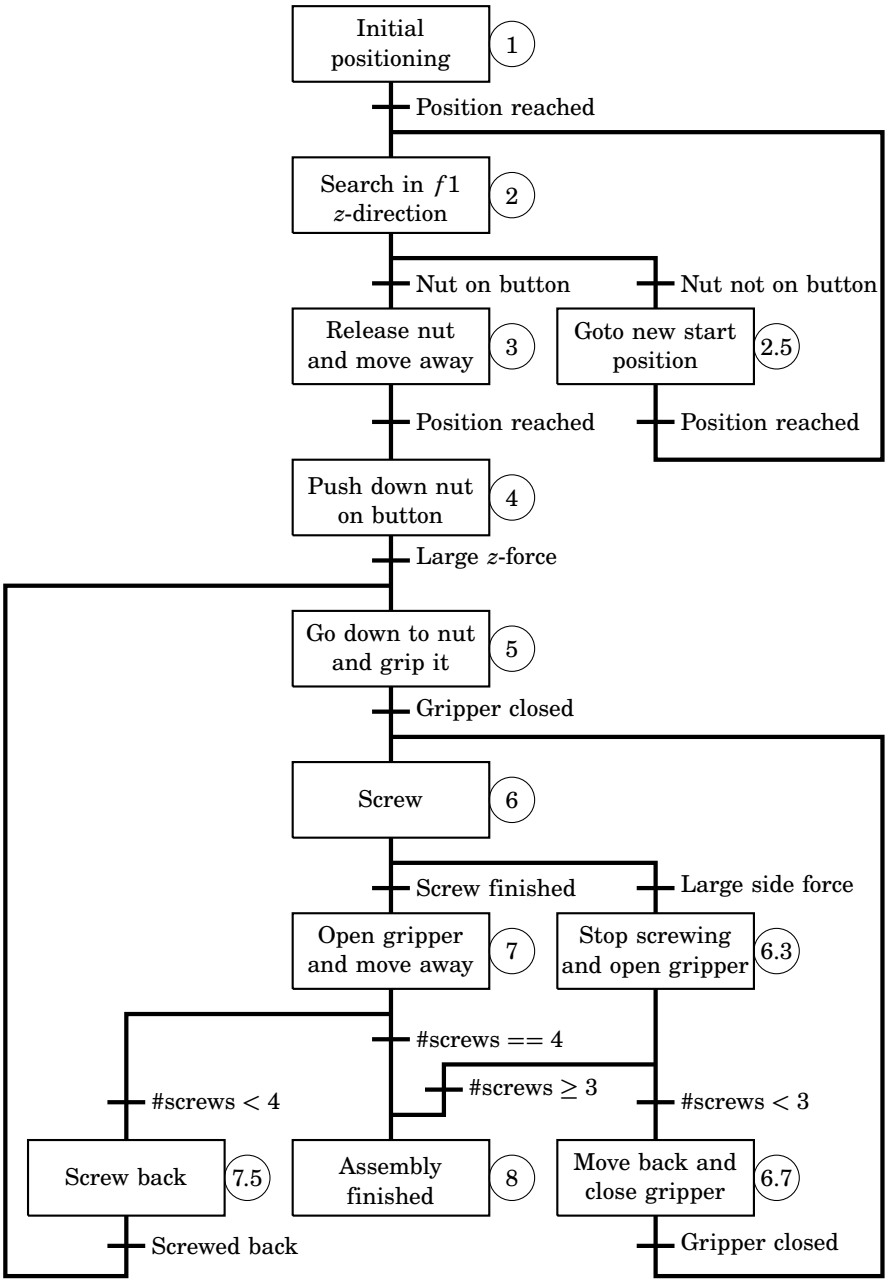


Figure 3.14 The state machine used for modeling the assembly strategy for the nut screwing skill. The expression #screws refers to the number of revolutions screwed.

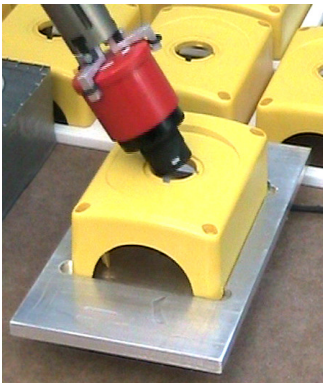


Figure 3.15 Putting the red button on the yellow box, a peg-in-hole operation.

released, the robot arms were slightly moved apart, and then the nut was gripped again, and the screwing continued. This set of actions was another example of an automatic error recovery strategy, and it usually led to a better grip.

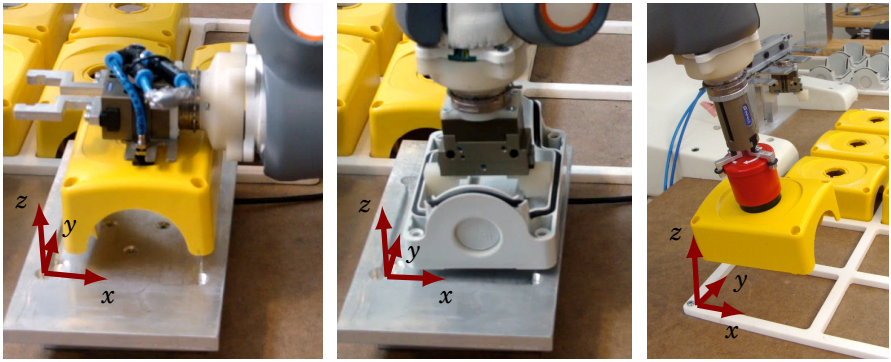
The arm holding the button was controlled to be still, as a motion of this arm might introduce disturbances to the screwing, due to errors in the kinematic models of the robot arms.

Snapfit The snapfit skill was presented in Sec. 3.2.

Peg-in-hole of red button in yellow box Putting the red button in the yellow box was a peg-in-hole operation. In the approach phase the button was tilted as in Fig. 3.15. The tilted orientation gave the button a more pointed contact surface, and hence less position accuracy was needed to hit the hole in the yellow box.

Once contact was made, the button was pressed downward and the forces in the radial directions of the hole were controlled to zero, in order to center the button in the hole. The button was then tilted until the center axis was in the vertical direction and the button slid down in the hole due to force control in the vertical direction. The orientation of the button was represented as a quaternion and the reorientation was performed by turning the button toward a target quaternion. For more details about how quaternions can be used together with the iTaSC framework, see [Stolt et al., 2012].

Putting a box in a fixture Putting a yellow or gray box in the fixture was performed by three guarded search motions in sequence as described in the bullet list below, see also Figs. 3.16(a) and 3.16(b).



(a) The yellow case. (b) The gray box. (c) Insertion into the intemediate storage.

Figure 3.16 Putting a box in a fixture skill.

1. The box is moved to an initial position close to the fixture, using position control. The initial position is slightly further in the positive directions of the x-, y- and z-axes than the final position, as shown in Fig. 3.16.

2. The box is moved in the negative z-direction until a downward force is detected on the fixture.

3. A force controller is used to keep pressing the box in the negative z-direction and the box is moved in the negative x-direction until a force in the negative x-direction is measured on the fixture.

4. A force controller is used to keep pressing the box in the negative x-direction on the fixture and the box is moved in the negative y-direction until a force in the negative y-direction is measured on the fixture.

5. The box is now assumed to be in the slot on the fixture. The box is released and the robot is moved away.

The same strategy could also be used for inserting the yellow case with the button in the intermediate storage. This operation was, however, started in a tilted position to minimize the contact surface between the yellow case and the wooden table such that large friction forces were avoided, see Fig. 3.16(c), and a reorientation to the horizontal position was made in the end of the sequence. As no force sensor was available for this operation, force estimation as described in Chap. 8 was used. Due to compliance in the gripper, the robot moved back 2 mm after the estimated force had exceeded the threshold level used for detecting a contact.

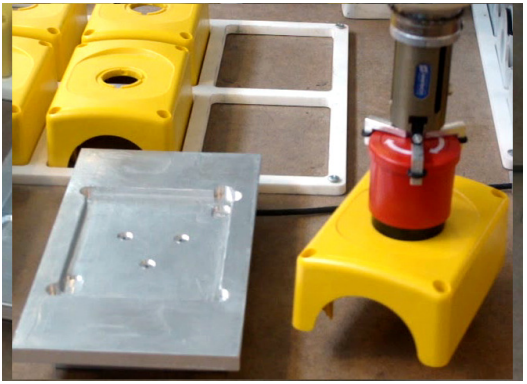


Figure 3.17 Example of initial position for the align box skill. The orientation of the yellow case in the gripper is unknown after screwing due to sliding in the gripper.

Aligning the box after screwing The orientation of the yellow case was unknown after the screwing, as sliding in the grip frequently occurred. An aligning phase against the fixture was therefore performed. A simple strategy would be to start from the position displayed in Fig. 3.17 and then perform a search motion towards the sensor until contact was established. The aligning against the fixture could then be performed by controlling the torque around the axis going through the button and the gripper to zero. This solution was, however, difficult to make fast, as the torque controller had to be well damped to be stable and therefore very slow.

To make the procedure faster, the following strategy was applied

1. Goto start position (beside the fixture as in Fig. 3.17)
2. Search for contact against fixture
3. If the distance to the fixture is not small enough, go back and rotate a certain angle and go back to previous step
4. Rotate to make contact with other end of case (rotation direction determined from measured torque)
5. Rotate to middle position
6. Search for contact against the fixture
7. Control torque to zero

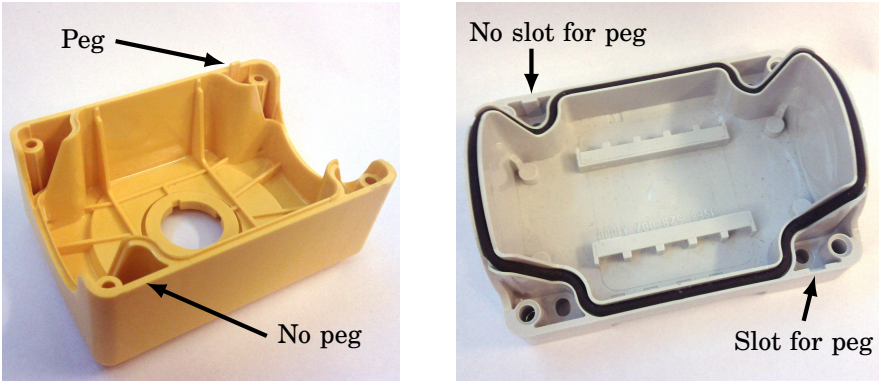


Figure 3.18 Photos showing the small peg on the yellow box and the matching slot on the gray box.

The skill has two extra phases compared to the simple strategy. The first one was used to quickly get to an orientation similar to the one in Fig. 3.17, i.e., the configuration where the short side of the yellow case was facing the fixture was ruled out. The second phase was used to get a start position for the torque control phase that was really close to being aligned with the fixture.

Check orientation of yellow case When the yellow case of the box was put on the gray bottom part, it may look like you could turn the yellow case 180 degrees around the center axis of the hole in the case, and it would still fit. This rotation was, however, not possible, due to a small peg in the yellow case and a corresponding slot in the gray part, illustrated in Fig. 3.18.

When the assembly was performed, it was assumed that the orientation of the gray part was known in its pallet. After the screwing and aligning of the yellow box (described in the previous subsections) it was not known if the yellow box was in its desired orientation or if it was turned 180 degrees.

To determine the orientation of the yellow case, it was put in contact with the fixture. The box was then moved in the positive y-direction, coordinate frame illustrated in Fig. 3.16(a), and if a force in the positive y-direction was detected by the force sensor, the peg was in the corner with the largest x and y coordinate. If the box could be moved a given distance without any detected force in the y-direction, the peg was in the corner with the smallest x and y coordinates.

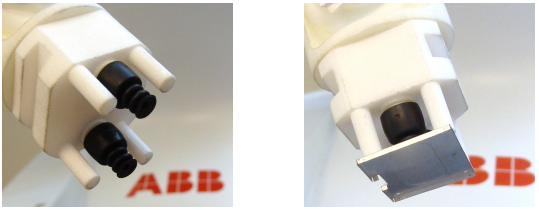


Figure 3.19 The vacuum gripper used in the shield-can assembly scenario. Left: Unloaded gripper. Right: Shield can in the gripper.

3.4 Shield can use case

This use case is a subassembly of a mobile phone. A 'shield can' (metal lid) should be assembled onto a printed circuit board (PCB), the parts can be seen in Fig. 3.20. The shield can should be pressed onto a socket on the PCB. There are no tolerances between the shield can and the socket, and the shield can will therefore have to be deformed to fit. The parts involved are small and fragile, and the assembly therefore has to be performed with care not to break anything.

Robot tooling

To make it possible to perform the mobile phone assembly, special tooling had been produced. A fixture had been designed for keeping the PCB in position. A suction tool was used to grasp the shield can, see Fig. 3.19. The maneuverability in contact was good in the vertical direction (the $f2$ z-direction in Fig. 3.20), but worse orthogonal to this direction ($f2$ x- and y-direction in Fig. 3.20), since the shield can might slide.

Task modeling

A kinematic chain was used in the assembly task and it is illustrated in Fig. 3.20. A schematic description of the kinematic chain is given in Fig. 3.21.

- Object frame $o1$ is attached to the fixture holding the PCB. It is related to the world coordinate frame by a constant transformation.
- Feature frame $f1$ is attached to one of the corners of the socket on the PCB. It is related to $o1$ by a constant transformation.
- Feature frame $f2$ is attached to one of the corners on the shield can.
- Object frame $o2$ coincides with the flange frame of the robot. The transformation between $f2$ and $o2$ is fix.

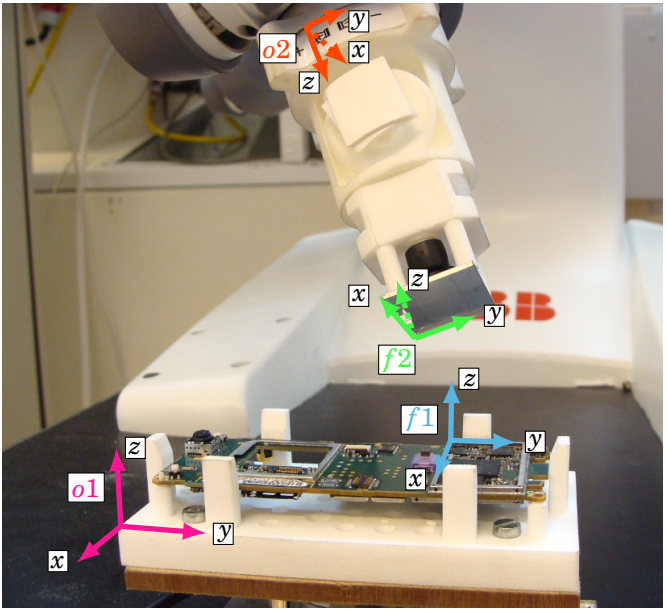


Figure 3.20 Illustration of the frames used in the shield can assembly task.

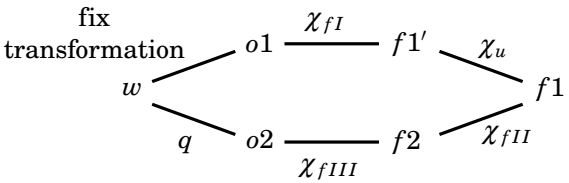


Figure 3.21 Schematic illustration of the kinematic chain used in the shield can assembly task.

The feature coordinates were all collected into the transformation between $f1$ and $f2$. The first three were Cartesian translations along the coordinate axes of $f1$ and the remaining reorientation was then parametrized by three Euler XYZ angles. The feature coordinates were divided into three groups depending on which frames they related to, according to:

$$\begin{aligned} \chi_{fI} &= (-) & o1 &\to f1 \\ \chi_{fII} &= (x, y, z, \varphi, \theta, \psi) & f1 &\to f2 \\ \chi_{fIII} &= (-) & f2 &\to o2 \end{aligned}$$

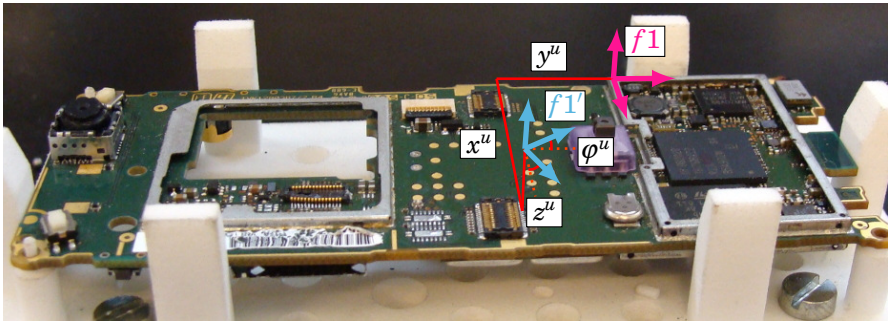


Figure 3.22 Illustration of the uncertainties in the shield can assembly task. The magnitude of the uncertainty is scaled up to make room for the frames and the labels.

Outputs were chosen to be all of the feature coordinates, according to

$$y_1 = x \quad y_2 = y \quad y_3 = z \quad y_4 = \varphi \quad y_5 = \theta \quad y_6 = \psi$$

Uncertainties Uncertainties in the task included the exact location and orientation of the fixture holding the PCB, and also how the shield can had been grasped. The first one could be modeled by introducing an uncertainty frame $f1'$. This frame represents the modeled position of the socket corner on the PCB, while $f1$ gives the real position. It was assumed that the fixture was mounted such that the PCB was placed in the horizontal plane, but the exact location and orientation in this plane were uncertain. This was modeled by introducing three uncertain translations and one uncertain reorientation angle (around the $f1$ z-axis), see Fig. 3.22. Similarly, in the grasp, the orientation around the z-axis and the translations along the x- and y-axes in frame $f2$ were uncertain.

The prior distribution of the uncertainty coordinates was assumed to have a standard deviation of a few millimeters and a few degrees, respectively. The only sensor information available was the contact force.

Assembly strategy The assembly strategy was designed such that the uncertainties were resolved in a robust way. A suitable strategy was to first find a corner of the socket with the shield can in a tilted position, see Fig. 3.20, by executing a sequence of guarded search motions, i.e., the search motions were stopped once the corresponding contact force was sensed. Once the corner was found, the shield can was rotated to what was estimated to be the correct orientation and it was then pressed onto the socket. When a certain force and torque were applied, the shield can could be considered to be mounted correctly.

By inspecting the PCB a suitable corner to try to find was the one where frame $f1$ was placed, see Fig. 3.22. The area in front of this corner

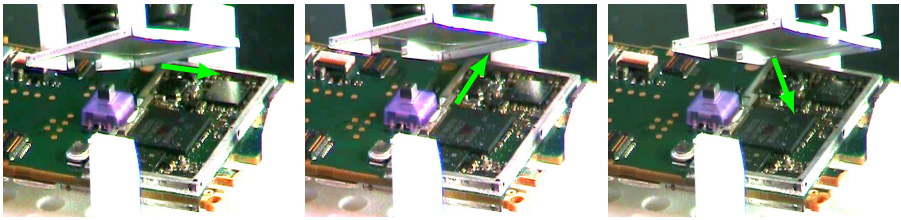


Figure 3.23 Snapshots from the shield can assembly sequence to illustrate how the corner is found. The arrows indicate in which direction the shield can is in contact. In the leftmost photo the robot is in state 5 and has sensed contact in the y-direction, in the middle photo the robot is in state 6 and has sensed contact in the x-direction, and in the rightmost photo the robot has found the corner.

was almost free of small edges that could lead to problems during the assembly. It was further large enough to be possible to find, considering the variance of the modeled uncertainties. A detailed assembly sequence is given below:

1. Pick up shield can from tray
2. Goto start position
3. Search for contact in negative $f1$ z-direction
4. Search for contact in positive $f1$ y-direction
5. Search for contact in negative $f1$ x-direction
6. Find corner of socket by yet another search in positive $f1$ y-direction (force control in x-direction)
7. Make a rotational search around the $f2$ x-axis and the $f2$ y-axis
8. Press shield can into position
9. Release shield can and move away with robot

An illustration of how the corner of the socket was found is given in Fig. 3.23.

3.5 Experimental results

The assembly use cases presented in Secs. 3.3 and 3.4 were implemented with industrial robots and experimentally evaluated. The cycle time and the robustness of the robot implementations were compared to that of a human performing the same tasks.

Table 3.1 Time taken for the skills in the emergency stop button assembly scenario with the YuMi implementation. The times for the position-controlled skills also include movements to and from the trays. The table also shows which of the arms was performing which skills.

Skill	Time	Right arm	Left arm
Pick yellow box	3 s	-	X
Put yellow box in fixture	2 s	-	X
Pick red button	3 s	X	-
Insert red button in yellow box	5 s	X	-
Pick nut	4 s	-	X
Lift and turn yellow box and red button	4 s	X	X
Put nut on thread	3 s	X	X
Screw nut	9 s	X	X
Movement from screwing to alignment	2 s	X	-
Align yellow box	2 s	X	-
Check orientation	2 s	X	-
Put in intermediate storage	4 s	X	-
Change tool	6 s	-	X
Pick gray box	2 s	-	X
Put gray box in fixture	2 s	-	X
Pick switch	2 s	-	X
Do snapfit	4 s	-	X
Put gray box and switch on table	1 s	-	X
Pick from intermediate storage	2 s	X	-
Put yellow box on top of gray box	3 s	X	-
Change tool	5 s	-	X

Emergency stop button

The complete assembly task was implemented with YuMi, the snapfit and the red button assembly skills were also implemented with the IRB140.

Timing of the assembly sequence—Robot implementations The total time for one cycle of the complete assembly task was 48 s. The measured time for the different skills are given in Table 3.1. Some of the skills could be performed in parallel, such as the picking of the red button and the picking of the yellow box, while some skills had to wait for other skills to finish using the shared resources, i.e., the fixture together with the force sensor.

Whereas the robustness properties for each one of the particular skills were quite good, it was rather worse for the entire task, as this required all skills to be successful. Common error cases included that the screwing skill failed, e.g., due to the robot losing the grip of the nut, or that the

nut ended up with a bad angle on the thread such that it got stuck. The success rate for the complete task was estimated to be 75 % when all skills were well tuned.

The snapfit and the red button skill were also implemented with IRB140. The snapfit assembly had a cycle time of 9 s and the red button assembly a cycle time of 11 s. The cycle times are thus more than doubled with IRB140 as compared to YuMi. The reason for this is that IRB140 is much heavier and stiffer than YuMi, which means that all search motions had to be much slower as the contact forces otherwise grew too large when contacts were established. The cycle time could be decreased by modeling and accounting for the uncertainties, as is described in Chap. 4, and doing this with the IRB140, the snapfit cycle time became approximately the same as for the YuMi implementation.

Timing of the assembly sequence—Manual assembly The cycle time for a human familiar with the task performing the assembly sequence was about 10–15 s, if the human was allowed to perform the assembly without restrictions. If the human had to perform the assembly using the fixture, in the same way as the robot, then the cycle time became 17 s. The measured time for each particular skill is given in Table 3.2. The human obviously does not need to change tool, and can also skip the alignment skill. Further, the human also waited with checking the 180 degree orientation of the yellow box until finalizing the assembly and putting it on top of the gray box.

Shield can

The shield can assembly task was implemented with YuMi. In an experiment, the task was executed 30 times with six different shield cans. The position of the shield can in its tray was varied between the executions, movements up to 3 mm, but not more than that the pins of the gripper supported the shield can when it was grasped. This variation was used to test the robustness of the task. The success rate was 83 % and the cycle time was 10 s. The error cases that occurred was that either the corner of the socket was not correctly detected, or that the shield can was not completely pressed onto the socket. A human who performed the same task completed the assembly in 2–3 s.

The robot implementation described in the previous paragraph used force control to maintain established contacts during sliding search motions. The search speeds had to be slow for the force controllers to be able to control the contact force in a robust manner. An alternative implementation was to use force sensing only for detecting the contact positions and then use position control to control the contact position. This strategy will be possible to use if the uncertainties in the orientation of the PCB in the

Table 3.2 Time taken for the skills in the emergency stop button assembly scenario with the human.

Skill	Time	Note
Pick yellow box	1 s	
Put yellow box in fixture	1 s	
Pick red button	1 s	
Insert red button in yellow box	1 s	
Pick nut	2 s	
Lift and turn yellow box and red button	2 s	
Put nut on thread	0 s	Done while turning
Screw nut	4 s	
Movement from screwing to alignment	0 s	Skipped by human
Align yellow box	0 s	Skipped by human
Check orientation	1 s	
Put in intermediate storage	2 s	
Change tool	0 s	Not necessary
Pick gray box	1 s	
Put gray box in fixture	1 s	
Pick switch	1 s	
Do snapfit	1 s	
Put gray box and switch on table	1 s	
Pick from intermediate storage	1 s	
Put yellow box on top of gray box	1 s	
Change tool	0 s	Not necessary

fixture are small. The assembly task was executed using this strategy in Chap. 8 (using force estimation instead of a force sensor), and the cycle time could then be decreased to 3.5 s.

3.6 Discussion

This chapter has described how assembly tasks can be specified and executed. Some skills could be position-controlled, but other skills needed to be performed using force-control. The need for force-sensing thus confirms that the approach to robotic assembly taken in this thesis using force control was appropriate.

Based on the experiments performed, it can be concluded that a skilled human assembly worker can perform assembly with a lower cycle time than a robot. For the emergency stop button assembly task, the human was approximately 3–4 times faster than the robot. There are possibilities to speed up the robot implementation by, e.g., optimizing trajectories for free space motions and minimizing the time that the arms have to wait for

each other during the assembly sequence. It is, however, not reasonable to believe that the cycle time can be made as fast as the human with the current workcell, i.e., with the only external sensor being the table-mounted force sensor. The human has several superior advantages, one is the vision feedback coming from the eyes. This makes it possible to start all assembly operations much closer to the final configuration than the robot was able to. For the force-controlled skills, the human uses force/tactile feedback with a strategy that is similar to that of the robot, but the human force control capabilities are much better than those of the robot.

The robot performs the screwing in 9 s, and the human in 4 s. The human hand is really much more suited for this kind of task than the gripper used for the robot, as the human uses a finger to spin the nut to efficiently fasten it. If the robot would not have needed to screw back after each revolution, because of limits of the robot joints, then the time needed for the screwing would be more than halved, and thus comparable to that of the human. It could, for instance, also be possible for the robot to use a specialized screwing tool in the workspace, then it could actually be possible to do it faster than the human.

An advantage with a robot is that it can work 24 hours per day, and 7 days per week, and it will in this time have time to complete about 12,600 stop buttons (with a cycle time of 48 s). This can be compared to a human that works 8 hours per day, and 5 days per week, that will be able to complete 9,600–14,400 stop buttons (with a cycle time of 10–15 s). In this sense, the current robot implementation has a cycle time comparable to that of a human assembly worker.

The analysis so far has been made with the assumption that nothing goes wrong. Some of the skills have been extended, such that they can detect and recover from common errors, but there are still many errors that are not handled by the system. For instance, the screwing skill is very error prone, as the fast screwing speed makes it crucial to detect bad grips very fast, and the estimated force is not always this fast and the result is that the robot stops. In the current setting it must be manually restarted, but in the future of course it could be automatically taken care of. Producing over 10,000 stop buttons per week with the current implementation is therefore unrealistic, but by adding more sensors to the system, e.g., wrist-mounted force sensor and vision system, together with an easy way for an operator to include detection of and recovery strategies from common errors. By handling errors, the system will become less error prone over time, and then it may be realistic to reach the robustness needed to be able to compete with the human assembly workers.

The control in this thesis was both performed on the velocity and the acceleration levels. The control performance for the different control

modes were similar, and with the available research interface, the acceleration level specification was not really necessary. It would, however, be useful to know the accelerations if a dynamic model of the robot was available, as dynamic torque feedforward then could be used to increase the control performance.

Assembly experiments were performed both using YuMi and IRB140. For the IRB140, the velocities when searching for contact had to be slow, as the stiff and heavy robot structure led to that forces were quickly built up when contacts were established, with risk of damaging equipment. For YuMi, on the other hand, the light-weight robot resulted in a lower inertia, and together with the weaker and somewhat compliant structure, much larger search velocities could be used without getting too large contact forces. The YuMi robot was in this sense forgiving during contact operations, and thus more suitable for small parts assembly than the IRB140.

Specifying the assembly tasks considered in this chapter can not be considered as being easy. Expert knowledge is required, which is something that hampers real industrial applications. Teaching a human to perform these assembly tasks is much easier, and for the robot to be competitive, the teaching phase must be simplified. One approach for doing this is to perform the task specification on a higher level, similar to how a human is instructed. This approach is the topic of Chap. 6. The teaching phase can also be simplified using lead-through programming, i.e., teaching skills to the robot by manually guiding it, which is the topic of Chap. 10.

3.7 Conclusions

A framework for robotic assembly was presented. Constraint-based specification of force-controlled skills based on the iTaSC framework were used, both on the velocity and the acceleration level. It was further reported how standard position-based control using an industrial controller was integrated with an external controller performing force control.

The task specification framework was illustrated in detail for two different assembly tasks. Results from implementation of these assembly tasks were reported, and the performance was compared to that of a human. It was concluded that a human was faster than the robot, but the robot could reach the same level of productivity as the human by working around the clock. The robustness of the robot implementation is still, however, not sufficient for industrial usage, and further work must be spent on simplifying addition of error detection and recovery strategies.

4

Uncertainty Estimation in Robotic Assembly

4.1 Introduction

Traditional position-controlled robots require a very structured environment to work well, as everything within the task must be known with a certain accuracy for the robot to be able to complete the task. The required accuracy depends on, e.g., the tolerances of the gripper and the parts involved in the task. A position-controlled system like this has difficulties in handling uncertainties in the tasks. Accurate fixtures and other task-specific solutions are used to minimize the uncertainties, so they are within the tolerances of the system, e.g., specialized procedures for gripping parts in exactly the same way every time. These implementations are quite time consuming and also inflexible, as much work usually is required when something in the task changes.

Relaxing the requirements on a structured environment directly introduces uncertainties in the task, e.g., positions of parts might no longer be exactly known and exact gripping might not be possible any more. To be able to cope with these circumstances, extra sensing might be needed, such as vision and force sensing. A systematic way of modeling geometric uncertainties and incorporating external sensing is provided in the constraint-based task specification framework (iTaSC) [De Schutter et al., 2007].

An introduction to the importance of taking uncertainties into account in robotics, along with the challenges that arise, is given in [Thrun, 2002]. An early framework for incorporation of uncertainties in the context of motion planning is described in [Donald, 1986]. Two examples of uncertainty management within the context of force control and compliant motion are [De Schutter, 1988] and [Lefebvre et al., 2005]. The first one considers how estimation of the motion of an object can be used to improve force control

against the same object, and the second is about how active sensing can be used to resolve uncertainties.

A method to handle uncertainties in assembly tasks is presented in [Xiao and Volz, 1989], where automatic replanning was performed when undesired contact situations were detected. Uncertainties of the sensor measurements were taken into consideration, which included position, velocity, and force. Another approach for handling uncertainty within the context of force-controlled assembly is presented in [Huang and Schimmels, 2003]. In that paper, uncertainty was handled by designing the force controller, an admittance controller, such that the motion resulting from contacts reduced the part misalignment. In this thesis, undesired contact situations and part misalignment were handled at the skill level, by the design of an appropriate state machine.

This chapter describes how uncertainties in robotic assembly can be modeled and resolved using force sensing. Two different uncertainties from the snapfit assembly scenario will be used as illustrating examples.

4.2 Modeling

Uncertainties are handled in iTaSC by assuming that the pose of some of the modeled frames are uncertain. Uncertainty coordinates, denoted by χ_u, are used to represent the directions in which the uncertainty is present.

The model update and estimation procedure (Sec. 3.2) is divided into two parts when uncertainty coordinates are introduced. First, the uncertainty coordinates are assumed to be known and constant when the feature coordinates are calculated. Then an estimator is fed with measurement data and the values of the uncertainty coordinates are updated. There is no general procedure for how to create such an estimator. Instead it has to be derived in each particular case.

The general goal with modeling and estimating uncertainties is to increase the performance of the task execution. Less uncertainty makes it possible to achieve a decreased failure rate, and it can also make it possible to decrease the cycle time of the task.

4.3 Example from snapfit assembly

The snapfit assembly scenario was described in Sec. 3.2, and it will be used to illustrate how uncertainties in an assembly task can be modeled and resolved. Uncertainties in the task include the exact location and orientation of the bottom box. These uncertainties can be modeled by introducing an uncertainty frame $f1'$. It represents the estimated position

and orientation of the frame $f1$, which is defined with respect to the actual position of the bottom box. The uncertainty coordinates are three translations and three Euler angles, i.e., uncertainties in all directions. The uncertainties can be resolved by using guarded search motions, i.e., motions that are velocity controlled in the search direction and stopped when a corresponding contact force is detected. Once contact is made, an estimate of the corresponding uncertainty coordinate will be available. When all guarded search motions in the assembly sequence have been performed, the system will have an estimate of the position of frame $f1$. After performing the assembly several times, the system will learn where the box usually is, i.e., it will learn a probability distribution of the position of frame $f1$. This knowledge can be used to decrease the cycle time, by increasing the speed of the guarded search motions when no contact is expected to be made, and use the slow nominal search speed when the expected contact position is approached.

The exact grasp of the switch is also assumed to be uncertain, and the y- and z-distances from $f2$ to $o2$ (Fig. 4.1) are therefore modeled as the uncertainty coordinates y^u and z^u. The distance in x is also uncertain, but it is small compared to the other distances and therefore considered to be known with sufficient accuracy.

An illustration of the uncertainty coordinates y^u and z^u is given in Fig. 4.1. As the only sensor available is a force sensor, the estimation can only be performed when the switch is in contact with the environment. The uncertainty can be estimated by performing a rotation in ψ (rotation around $f2$ x-axis, see illustration in Fig. 4.2) while keeping the switch in contact with the box (this corresponds to state 5 in the assembly sequence, see Fig. 3.4). If y^u and z^u were known exactly, the contact forces at the origin of $f2$ would remain constant during the rotation without any force control. In practice the contact forces will change and force controllers will modify the position and velocity references in the y- and z-directions to maintain the forces.

Let us assume that there is an estimation error $\tilde{y}^u = y^u - \hat{y}^u$ and $\tilde{z}^u = z^u - \hat{z}^u$, illustrated in Fig. 4.2, where $\hat{y}^u$ and $\hat{z}^u$ are estimates of y^u and z^u, respectively. These estimation errors give rise to attempted rotations around the origin of frame $f2'$ instead of around the origin of $f2$. Since the contact is force controlled the actual rotation will, however, be made around the origin of $f2$, and the velocity of $o2$ will be

$$v = [\dot{\psi}, 0, 0]^T \times [x_{dist}, y^u, z^u]^T = [0, -\dot{\psi} z^u, \dot{\psi} y^u] \tag{4.1}$$

where $\times$ denotes the vector cross product. This assumption holds only if the force controllers manage to control the contact forces to the reference values, i.e., the force controllers have to be fast, the estimation errors small, or the rotational search speed low.

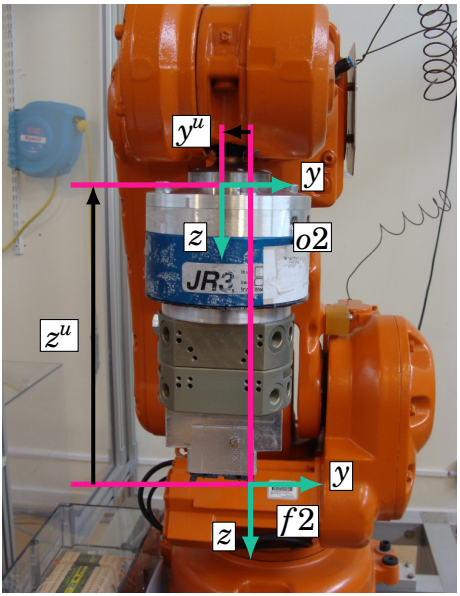


Figure 4.1 Illustration of the uncertainty coordinates y^u and z^u between the two frames $f2$ and $o2$.

The assumption described in the previous paragraph can be used to set up a dynamical model for y^u and z^u, according to (4.2), where the state $s = [y^u, z^u]^T$ and the measurement $m = v$ is the linear velocity of frame $o2$, g and n are Gaussian noise.

$$\begin{cases} \dot{s} = g(t) \\ m = -\dot{\psi} s + n(t) \end{cases} \tag{4.2}$$

A Kalman filter [Kalman, 1960] can be used to estimate s. A discretized model of (4.2) is (4.3), where k denotes the sampling index. The system matrices are defined in (4.4). The noise covariances are assumed to be given by Eq. (4.5).

$$\begin{cases} s(k+1) = As(k) + w(k) \\ m(k) = C(k)s(k) + e(k) \end{cases} \tag{4.3}$$

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad , \quad C = \begin{bmatrix} 0 & 0 \\ 0 & -\dot{\psi} \\ \dot{\psi} & 0 \end{bmatrix} \tag{4.4}$$

$$\begin{aligned} E\left[w(k)w^T(k)\right] &= Q(k) \\ E\left[w(k)e^T(k)\right] &= 0 \\ E\left[e(k)e^T(k)\right] &= R(k) \end{aligned} \tag{4.5}$$

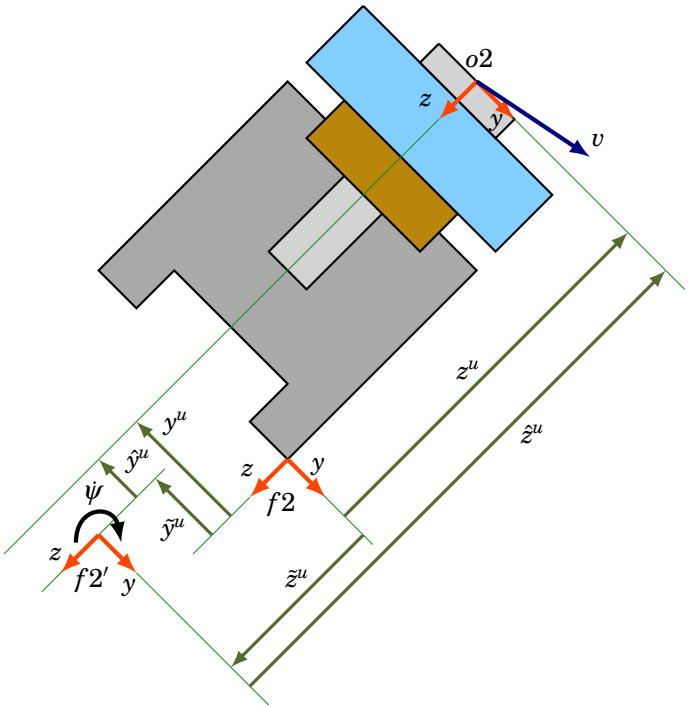


Figure 4.2 Illustration of the uncertainty coordinates y^u and z^u.

A Kalman filter for the model (4.3) is given by (4.6)–(4.12).

$$\hat{s}(k|k-1) = A\hat{s}(k-1|k-1) \tag{4.6}$$
$$P(k|k-1) = AP(k-1|k-1)A^T + Q(k) \tag{4.7}$$
$$\tilde{m}(k) = m(k) - C(k)\hat{s}(k|k-1) \tag{4.8}$$
$$S(k) = C(k)P(k|k-1)C^T(k) + R(k) \tag{4.9}$$
$$K(k) = P(k|k-1)C^T(k)S^{-1}(k) \tag{4.10}$$
$$\hat{s}(k|k) = \hat{s}(k|k-1) + K(k)\tilde{m}(k) \tag{4.11}$$
$$P(k|k) = (I - K(k)C(k))\,P(k|k-1) \tag{4.12}$$

4.4 Experimental results

The estimation schemes described in the previous section were implemented and evaluated experimentally. The distribution of the position and orientation of the bottom box was learned by performing the assembly

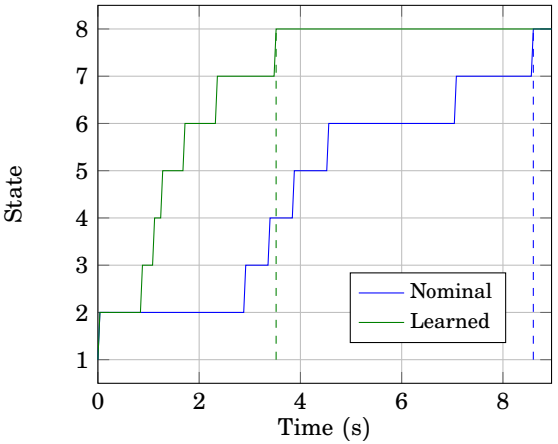


Figure 4.3 Time evolution of the state sequence from the snapfit assembly task with IRB140 with and without learning of the position and orientation of the bottom box. The vertical dashed lines indicate when the assembly operations finish.

operation a number of times, using a slow search speed that was tuned to be as fast as possible without generating too large forces when contacts were established. In the experiments, the box was placed in a fixture, and the variation of its position between executions was therefore very small, and it was sufficient to perform the assembly about 5–10 times to learn the distribution. The experiment was first performed with the ABB IRB140. Here it was possible to increase the search speed 4 times in areas where no contact was expected, and this decreased the cycle time from 8.6 s to 3.6 s. The states over time with and without learning are displayed in Fig. 4.3. The same experiment was also performed with the ABB YuMi. There was much less benefit of using the learning strategy now. The reason is that the slow searches could be made much faster with YuMi, as the robot has a lower inertia and is much less stiff, which results in that forces do not build up as quickly as when the IRB140 is used. As compared to the IRB140 implementation, the slow search speeds with YuMi could be made six times faster, i.e., even faster than the increased search speeds with the IRB140. Increasing the free space search speeds in the YuMi implementation then only resulted in a negligible decrease in cycle time.

The estimation scheme for the gripping uncertainty was implemented with the IRB140 in a scenario similar to the snapfit assembly. Initial contact was searched for in the z-direction, when this contact was established it was force controlled and a new search in the y-direction was performed.

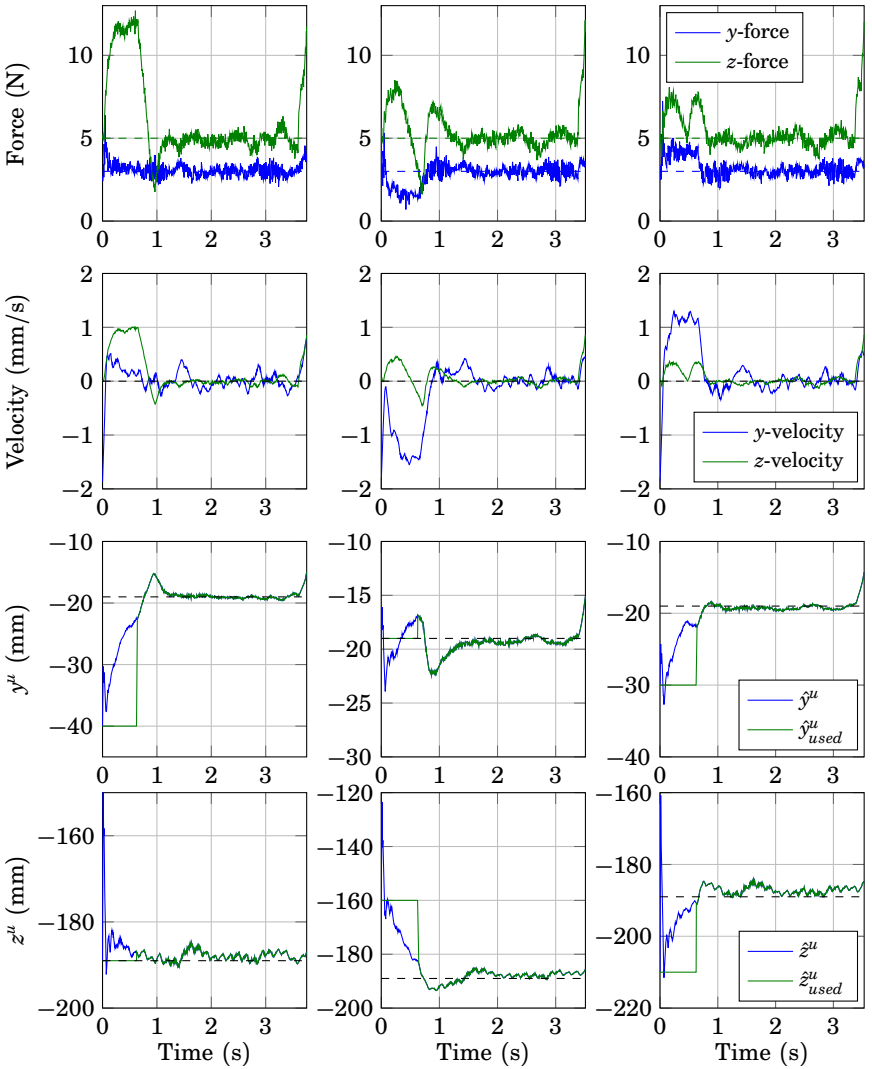


Figure 4.4 Experimental data from the three gripping uncertainty estimation experiments. The uppermost row of diagrams shows the y- and z-forces, measured data with solid lines and references with dashed lines. The second row of diagrams shows the y- and z-velocities, i.e., the force controller outputs, which should be equal to zero (indicated by black dashed lines). The lowermost rows of diagrams finally show the time evolution of the uncertainty coordinate estimates (blue), the dashed lines indicate the correct value, and the green lines show the estimate used for control, i.e., the initial guess used until the estimation covariance had settled down.

Once contact was established also in this direction, it was force controlled and a rotation in ψ (around the $f2$ x-axis) was started. Simultaneous to the rotation, the estimator was also started. Three different initial guesses were used and the result is displayed in Fig. 4.4, where each column of diagrams corresponds to one initial guess.

The state and measurement covariances (4.5) were chosen to be constant and diagonal. The covariances were manually tuned to give a satisfactory estimation performance.

All three experiments started with relatively large force transients. The explanation for this was the inaccurate initial guess of the uncertainty coordinates, which meant that the rotation was not performed around the contact point. A contribution to the transient was also the newly established contact, the force controller needed some time to settle down after making contact as the environment was quite stiff. This latter explanation for the transient behavior was not modeled, and it therefore resulted in some temporary estimation errors, which can be seen in the beginning of the resulting estimates for the three experiments. The initial behavior was also caused by the fact that the initial state covariance was chosen to be large, as it was assumed that the initial guess was poor. The actual estimate of the uncertainty coordinates used during the execution was the initial guesses, at least until the Kalman filter had converged. This was considered to have happened when the rate of change of the covariance had decreased below a threshold.

The estimation performance was good, as estimation errors of up to 30 mm converge in less than 2 s for all three experiments. Once the estimates have converged, the forces track the references in a satisfactory way, 3 N in the y-direction and 5 N in the z-direction. The velocity corrections made (by the force controllers) can also be seen to become very small. With the uncertainty coordinates known, it is possible to increase the assembly speed, as the force controllers now have to do less corrections. The risk for errors occurring is also decreased.

The y^u-estimate can be seen to increase fast in the end of each experiment, this was caused by the rotational search making contact with the other end of the switch. The switch then had two contact points, i.e., a contact situation that was not modeled.

4.5 Conclusions

A method of modeling uncertainties in robotic assembly based on the iTaSC framework was described. The methodology can be applied to any geometric uncertainty, as long as it is possible to relate it to the measurements from a sensor. The method requires the user to manually derive

an estimator for each uncertainty to resolve, including a measurement model. Making it completely automatic, i.e., generating an estimator to an uncertainty coordinate specified by the user, is, however, difficult and was not within the scope of this work.

Two different types of uncertainties in the snapfit assembly scenario were modeled and resolved. By learning the approximate position of the bottom box, it was possible to speed up the search motions and decrease the cycle time. Experiments with two different industrial robots were performed, and it was seen that the strategy was especially beneficial when used with a stiff robot, where the nominal search speeds have to be slow when the uncertainties are large. The second considered uncertainty was a gripping uncertainty that was resolved using a Kalman filter.

5

Detection of Contact Force Transients

5.1 Introduction

Industrial robots are usually position-controlled, i.e., controlled to follow predefined trajectories. The robots do the trajectory following very well, with high speed and very good repeatability. But not all types of tasks are suitable for this control strategy. Tasks that require interaction with the environment, such as assembly, would require a very high accuracy of the robot and the calibration of its workcell to be able to accomplish the tasks using position control. Small uncertainties in position may lead to very large forces and potentially damaged equipment. An alternative strategy is to introduce additional sensors, such as a force sensor. Uncertainties can then be compensated for by sensing the contact forces.

A force-controlled assembly task can be implemented as a sequence of simple motions, usually in the form of search motions that are used to resolve uncertainties in the task. Examples of this strategy was presented in Chap. 3. The condition for switching between two such simple motions is that some event occurs, e.g., that a new contact situation is established. These events can often be detected by using a threshold on the measured force/torque data. What really is detected is a transient in the measured data. In many cases, the events can be detected from other transients as well, occurring before the force/torque build up detected by the threshold. This effect can be exploited to reduce the total assembly time. This chapter considers the problem of detecting transients in force/torque data in the context of force-controlled assembly, but it could also be applied to any other signal in a different context. A systematic approach for training machine-learning based classifiers will be used to accomplish this. The snapfit assembly scenario presented in Secs. 3.2–3.3 will be used as an experimental case.

Machine learning approaches within assembly have previously been used in [Rodriguez et al., 2010], where a single-axis force sensor was used to detect failures in an assembly scenario. A somewhat different approach was applied in [Di Lello et al., 2012], where a Hierarchical Dirichlet Process Hidden Markov model was used to monitor an assembly task, for detection of errors. Another approach to monitoring assembly tasks was presented in [Rojas et al., 2012], where a hierarchical taxonomy was used to monitor a snap assembly task. The force/torque signatures were investigated with respect to relative changes in several different layers that could be used to discriminate nominal behavior from errors. Machine learning has also been used for detecting failures, e.g., [S. Cho et al., 2005] and [Hsueh and Yang, 2008], where support vector machines were used to detect tool breakage in milling processes.

Assembly scenario

The assembly scenario considered in this chapter is the snapfit assembly scenario, presented in Secs. 3.2–3.3. The assembly sequence is controlled by a state machine, with force/torque measurements triggering state transitions. Simple threshold detectors for some force or torque channels can be used for all transitions. However, the overall assembly time can be decreased by instead detecting transients in the force/torque data. Another positive effect is that the magnitude of the forces in the task can be decreased, as the largest forces usually relate to the thresholds used, which usually need to be large to get robustness. In state number 6 (see state sequence in Figs. 3.4 and 3.5), the switch slides along a surface to find the slot and the search is finished when contact is established with an edge. A possible improvement is to instead detect the transient arising from the switch sliding into the slot, which happens before contact with the edge is established. This motion will be called the slide motion in the rest of the chapter. Another improvement is for the state where the switch is snapped into place (state number 7), where the transition event can be detected from the snap transient, instead from when contact is established with the bottom of the box. This motion will be called the snap motion.

5.2 Procedure

To get a classifier for a transient from an assembly sequence, several steps have to be taken. First, data have to be gathered and preprocessed, then the classifier has to be trained. The procedure is described in this section, and the classifiers considered are introduced. For method description, the snap motion transition is used as an example.

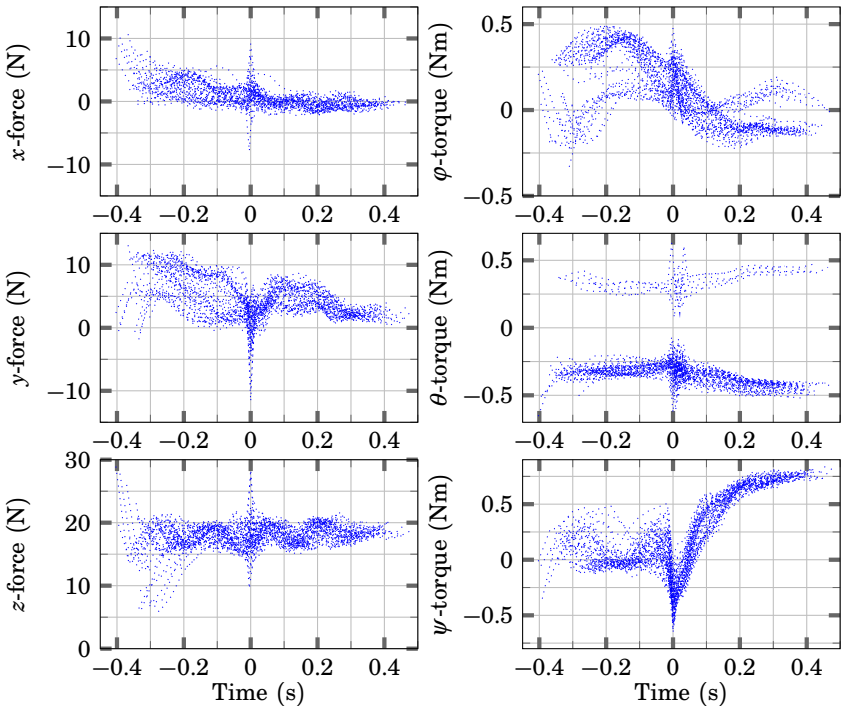


Figure 5.1 The recorded data from the snap motion, in total 60 different executions with 12 different switches. To increase the readability, all sequences have been shifted such that the transient occurs at $t = 0$ s. The force/torque directions refer to the feature coordinates defined in Sec. 3.2.

Data acquisition

In a production setting, each emergency stop button that the robot assembles will have new components. To mimic this scenario, it would therefore be a benefit to have a large number of different switches available, recording one assembly operation from each one of them. In the current setup, however, only 12 switches were available, and only using them once would give a rather small training data set. Therefore, five recordings were made for each switch, and the total number of recordings made was thus 60.

The training data were divided into two halves, one to be used for training and one to be used for validation. The division was made such that the training and the validation data set did not contain any recordings from the same switch.

The transients considered, during the slide motion and the snap motion, were manually marked in each recording. The force/torque data from

the recorded snaps are displayed in Fig. 5.1. For the human eye, the easiest way to recognize the snap is probably to look in the ψ-torque data. It might seem to be an easy task to detect a snap in these data, but the classifier should be able to do it with only a subset of the data available. The smaller subset, the better, as that would decrease the computational burden, and also decrease the delay from when the snap occurs until the detection is done. The previously used threshold detector was that the ψ-torque should exceed 0.8 Nm.

Data pretreatment

The data contained six different channels of force and torque data, i.e., the three force directions and the three torque directions as displayed in Fig. 5.1. A subset of these channels were used by the classifier. A number n_{pre} of samples before the transient and a number n_{post} of samples after the transient were taken from the channels that were used and were put after one another in a vector, which will be called a sequence. If the number of channels used was n_{ch}, then a sequence will be a vector in $\mathbb{R}^{n_{ch} \cdot (n_{pre}+1+n_{post})}$. The job of the classifier is thus to determine if a sequence contains a transient or not.

Before forming the sequence vector, the mean value was removed from each channel. This should make the classifier independent to any offset force/torque, e.g., the offset seen for some of the executions in the θ-torque in Fig. 5.1. The data were also prescaled such that all force/torque components got approximately the same magnitude. This should make the problem of training the classifiers better numerically conditioned.

Data for training

Every recording contained one interesting transient and a lot of background sequences, i.e., data not containing a transient. The number of background sequences in each recording is approximately equal to the length of the recording, which was about 100 for the snap transient example. Using all background sequences would give an unreasonably large training data set. Instead, a number of N (chosen to be 20) randomly selected background sequences and the sequence containing the transient were used from each recording. These selected sequences were used for training the classifier, and then it was validated against all background sequences in the recordings chosen for training. If any misclassified sequences were found, i.e., 'hard examples', these were included into the training sequences and the training of the classifier was performed once again. This procedure was iterated until no more misclassified sequences in the recordings chosen for training were found, not already included in the training sequences, or until a maximum number of iterations was

reached (10 was used). A similar approach for choosing training samples was used in [Dalal and Triggs, 2005], in the context of detecting humans in images.

There is a risk, however, that this strategy will lead to different classifiers depending on the initial choice of background sequences. To average this effect out, for every choice of parameters, the classifier was trained a number of N_{seeds}, usually around 10–20, each time with a different configuration of the random number generator used for choosing the initial background sequences. The score for the given parameters was then taken as the average of the N_{seeds} performed trainings.

Cost function

It might be more important not to make any false positive classifications than any false negative classifications, or the other way around. In the experimental scenario considered, classifying a background sequence as a transient would be worse than missing a transient, as backup classifiers based on threshold levels existed. This asymmetry in the problem can be included by using the following cost function for training

$$J = cost_{FP} \cdot n_{FP} + cost_{FN} \cdot n_{FN} \tag{5.1}$$

where $cost_{FP}$ and $cost_{FN}$ are the costs associated with false positives and negatives, respectively, and n_{FP} and n_{FN} are the number of false positives and false negatives.

Classifiers considered

Four different classifiers were applied to the data, a simple least-squares classifier, two different versions of support vector machines, and a boosting classifier. A detailed description of the methods can, e.g., be found in [Bishop, 2006], and a summary of them can be found below.

Least-squares method The first classification method used was based on the least-squares (LS) method. The model used was

$$y(x) = w^T x + w_0 = \underbrace{\begin{bmatrix} w^T & w_0 \end{bmatrix}}_{\tilde{w}^T} \underbrace{\begin{bmatrix} x \\ 1 \end{bmatrix}}_{\tilde{x}} = \tilde{w}^T \tilde{x} \tag{5.2}$$

where x denotes the data sequence, $\tilde{w}$ the parameters, and y should be 1 for a transient and -1 for the background. The classifier is based on finding a hyperplane that separates the transients from the background.

All training data can be described by

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} \tilde{x}_1^T \\ \tilde{x}_2^T \\ \vdots \\ \tilde{x}_n^T \end{bmatrix}}_{\tilde{X}} \tilde{w} \tag{5.3}$$

and by using a sum-of-squares error function

$$J_{LS}(\tilde{w}) = \sum_{i=1}^{n} c_i (y_i - \tilde{x}_i^T \tilde{w})^2 = (Y - \tilde{X}\tilde{w})^T C (Y - \tilde{X}\tilde{w}) \tag{5.4}$$

where C is a diagonal matrix describing the cost for the element i, the optimal $\tilde{w}$ is given by

$$\tilde{w} = \left(\tilde{X}^T C \tilde{X}\right)^{-1} \tilde{X}^T C Y \tag{5.5}$$

The cost matrix C was chosen such that $c_i = cost_{FP}$ if $y_i = -1$ and $c_i = cost_{FN}$ if $y_i = 1$. This gives an approximation of the cost function (5.1). Classification is performed by calculating $y(x)$, and the classification boundary is 0, i.e., $y(x) > 0$ means that x is classified as a transient.

Support vector machines The simplest form of support vector machines (SVM) uses the model (5.2), but instead of choosing the parameters based on the error function (5.4), the parameters are chosen such that the resulting hyperplane is the one with the largest margin to the different classes of data. The problem can be stated as the optimization problem

$$\begin{array}{lll} \underset{\text{over } w,\zeta}{\text{minimize}} & P \sum_{i=1}^{n} c_i \zeta_i + \frac{1}{2}\|\tilde{w}\|_2^2 & \\ \text{subject to} & y_i y(x_i) \geq 1 - \zeta_i & , \quad i = 1, \ldots, n \\ & \zeta_i \geq 0 & , \quad i = 1, \ldots, n \end{array} \tag{5.6}$$

where ζ_i are slack variables that allow for misclassifications, c_i is the cost for the misclassification, and $y(x)$ is defined as in (5.2). The cost is chosen as $c_i = cost_{FP}$ if $y_i = -1$ and $c_i = cost_{FN}$ if $y_i = 1$, i.e., an approximation of (5.1). The sum of the misclassifications is penalized by the parameter P. This parameter will control the trade-off between classifying everything correct versus having a large margin with possibly a few incorrect classifications. The optimization problem (5.6) is convex, and it can therefore be solved efficiently with guarantee of finding global minimum. This classifier will be called the primal SVM.

By considering the dual formulation of (5.6), it can be shown [Bishop, 2006] that the following problem is equivalent to solve

$$\begin{aligned} \underset{\text{over } a}{\text{minimize}} \quad & \sum_{i=1}^{n} a_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j y_i y_j k(x_i, x_j) \\ \text{subject to} \quad & 0 \leq a_i \leq Pc_i \quad , \quad i = 1, \ldots, n \\ & \sum_{i=1}^{n} a_i y_i = 0 \end{aligned} \tag{5.7}$$

The function $k(x_i, x_j) = x_i^T x_j$ is a kernel function, which can be replaced by any other positive definite kernel function.

The formulation (5.7) was used together with a Gaussian kernel, $k(x_i, x_j) = e^{-(x_i - x_j)^T (x_i - x_j)/l}$, where $l > 0$ is a parameter that corresponds to the width of the kernel. The classification function is given by

$$y(x) = \sum_{i=1}^{n} a_i y_i k(x, x_i) + b \tag{5.8}$$

Most of the a_i are zero, and the non-zero ones correspond to the support vectors x_i. The parameter b is calculated through

$$b = \frac{1}{N_M} \sum_{i \in M} \left(y_i - \sum_{j \in S} a_j y_j k(x_i, x_j) \right) \tag{5.9}$$

where S is the set of all support vectors, M is the set of indices of data points having $0 < a_i < Pc_i$, and N_M is the number of elements in M.

Boosting This is a classifier that uses the result from many simple classifiers, called weak classifiers, to make the final classification. The performance of the final classifier is usually significantly better than any of the weak classifiers. During training, weights are used to give data points that are hard for the weak classifiers to correctly classify more importance. The final classifier is a weighted average of the weak classifiers, which in turn has been trained with differently weighted training data. The algorithm used is called AdaBoost and was first proposed in [Freund and Schapire, 1996].

The weak classifiers considered were threshold detectors for one of the coordinate axes. For each weak classifier, all axes were considered, and the one giving the best separation, concerning the cost function (5.1) and the importance weights, was chosen.

Training procedure

The first part of the training procedure consisted in deciding which channels to use, the number n_{post} samples after the transient, and the number

of n_{pre} samples before the transient. This is a difficult task, as the available parameter space is quite large. A proposal is that it can be performed in three steps, by first selecting the channels to use, then the number of samples after the transient, and finally the number of samples before the transient.

The values of the parameters to be used can be found by choosing one classifier and evaluate the performance on the validation data for different parameter settings. It is also possible to use a combination of classifiers.

Varying the channels to use The first step was to vary the channels used. The values of n_{pre} and n_{post} were chosen to be large (30 was used), such that the limiting factor for the total information available was in the choice of channels. All possible combinations of channels were evaluated.

One choice for continuing from this step would be to choose the combination of channels giving the best result, but this would most likely mean that all channels should be chosen. That is undesirable, as the complexity of the classifier increases with the number of channels used. Instead, the best choices for one to four channels were chosen and used for the following steps.

Varying $\mathbf{n_{post}}$ The number of samples used after the transient should be kept at a minimum, not delaying the detection of the transient more than absolutely necessary. This was done by varying n_{post} from zero to a large value (30 was used), while fixing n_{pre} to a large value (30 was used). In other words, all information before the transient was kept while the amount of information after the transient was varied. The evaluation was based on the performance measure J_1, the sum of J defined in (5.1) plus a quadratic penalty term.

$$J_1 = J + p_1 n_{post}^2 \tag{5.10}$$

Varying $\mathbf{n_{pre}}$ The trade-off for the number of samples used before the transient is about performance versus complexity of the classifier. The parameter n_{pre} was varied from zero to a large value (30 was used), with n_{post} given by the choice from the previous step. The evaluation was similar as in the previous step, but now with the linear penalty $p_2 n_{pre}$. This should reflect that it is not as costly to have some more samples before the transient than after.

Training of other classifiers With all parameters concerning how much data the classifier should use decided, choosing the classifier specific parameters remained. Neither the LS nor the boosting classifier ($N = 50$ weak classifiers were used) had any specific parameters, but the SVM classifiers considered had some further parameters to tune. The primal

formulation has the so called misclassification penalty, P in (5.6). The search for the best choice of this parameter was performed in two steps. First the parameter space was coarsely gridded with logarithmic spacing. A classifier was trained in every grid point and evaluated against the validation data. The second step consisted in refining the grid around the best grid points from the first step and performing the training in the new grid points.

The dual SVM classifier had the width of the Gaussian kernel as an extra parameter, besides the misclassification penalty. The parameter space was therefore two-dimensional. The training procedure was performed in the same way as for the primal SVM classifier, but with an even coarser grid to begin with to reduce the computational load.

Implementation

The training procedure described in this section was implemented in Matlab. The training of the SVM classifiers was performed by using CVX, a package for specifying and solving convex programs [CVX Research, 2012; Grant and Boyd, 2008].

5.3 Experimental results

Robot system

The robot used in the assembly scenario was the ABB YuMi, a photo of the experimental setup is displayed in Fig. 5.2. An ATI Mini40 six degrees-of-freedom force/torque sensor was mounted on the table beneath the fixture for the bottom box. The sensor was used without any configured low-pass filter, and it was sampled at 250 Hz.

Snap detection

Three different choices of classifiers were used for the initial training phase, namely LS, boosting, and the combination of LS and boosting. These classifiers were used as they had no parameters to tune, and as they were both quite computationally cheap. The first parameter to be decided was which channels in the measured force/torque data to use. The result on validation data for all combinations of channels is displayed in Fig. 5.3. It can be seen that the variation in the performance decreases when the number of channels is increased, which is intuitive as more information becomes available. For the LS classifier, however, the performance seems to deteriorate when the number of channels becomes large. This is probably due to the classifier being over-fitted, and thus failing on validation data. The best choice for one to four channels was used for further training.

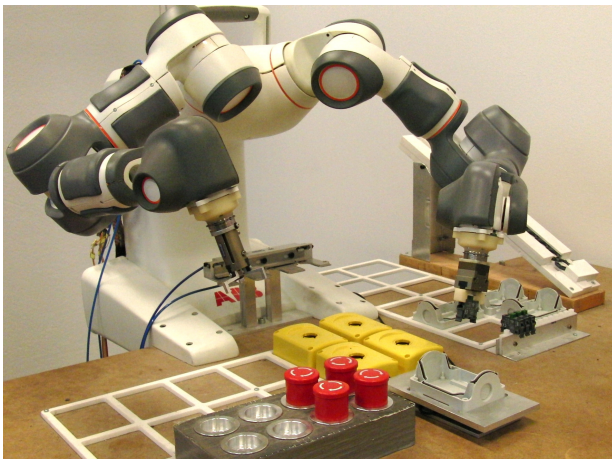


Figure 5.2 The experimental setup used for the experiments.

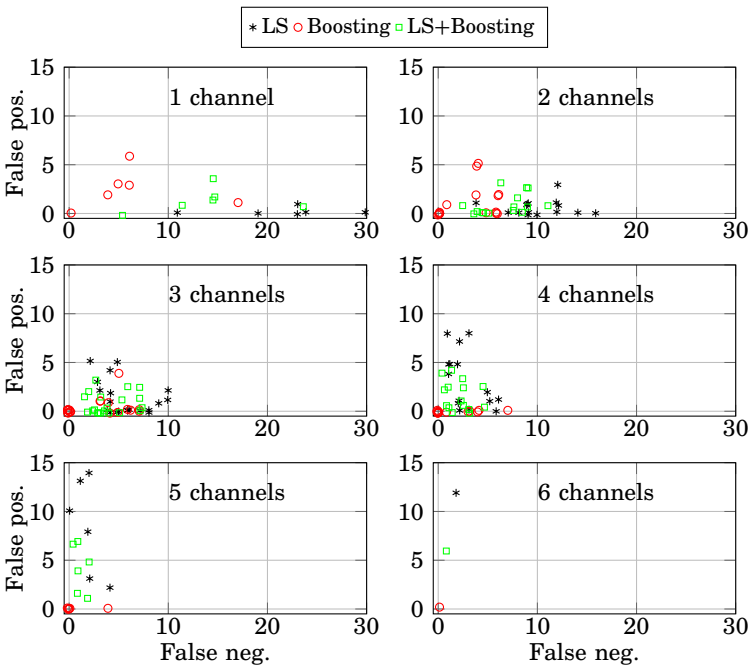


Figure 5.3 The classification results in the snap detection scenario when all different combinations of channels were tested. A small random perturbation has been added to every marker, such that it should be possible to see where multiple configurations led to the same result.

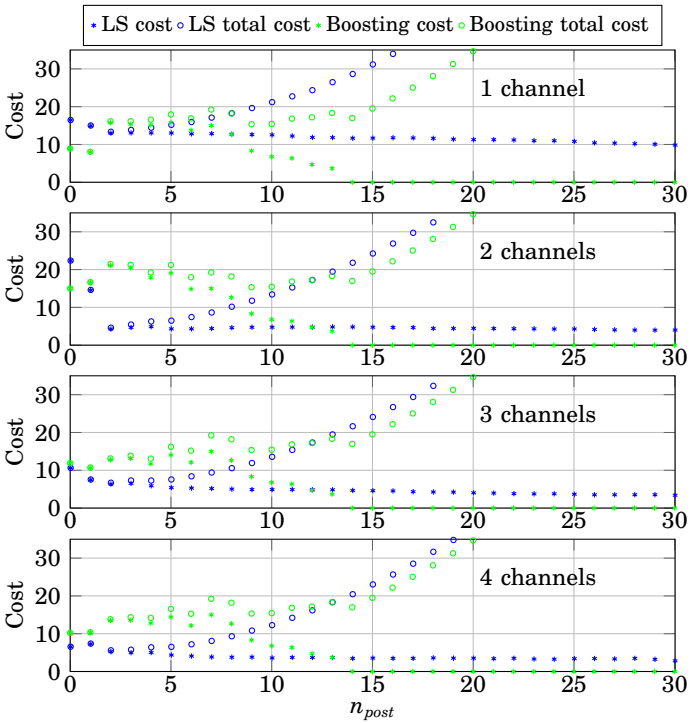


Figure 5.4 The result when the number of samples after the transient, n_{post}, was varied. The cost refers to the cost function (5.1) with $cost_{FP} = cost_{FN} = 1$. The total cost also contains the extra penalty term for large n_{post}.

The next parameter to decide was n_{post}, the number of samples to use after the snap. The result on validation data when this parameter was varied from 0 to 30 for the different choices of channel selections is displayed in Fig. 5.4. The training based on the combination of LS and boosting has been omitted from the figure to increase readability. The stars ($*$) show the cost according to (5.1), i.e., the number of misclassifications as $cost_{FP} = cost_{FN} = 1$, and the rings ($\circ$) the total performance measure (5.10). The constant p_1 was chosen as 0.09, and it was chosen such that it gave a quite low value for n_{post} for this case. The LS classifier behaves as expected, i.e., the performance increases with n_{post}. The boosting classifier, on the other hand, first shows a performance degradation before giving perfect classification for $n_{post} \geq 14$. For all cases the n_{post} given by the performance measure was a low value.

The final parameter to decide was n_{pre}. The result from varying this

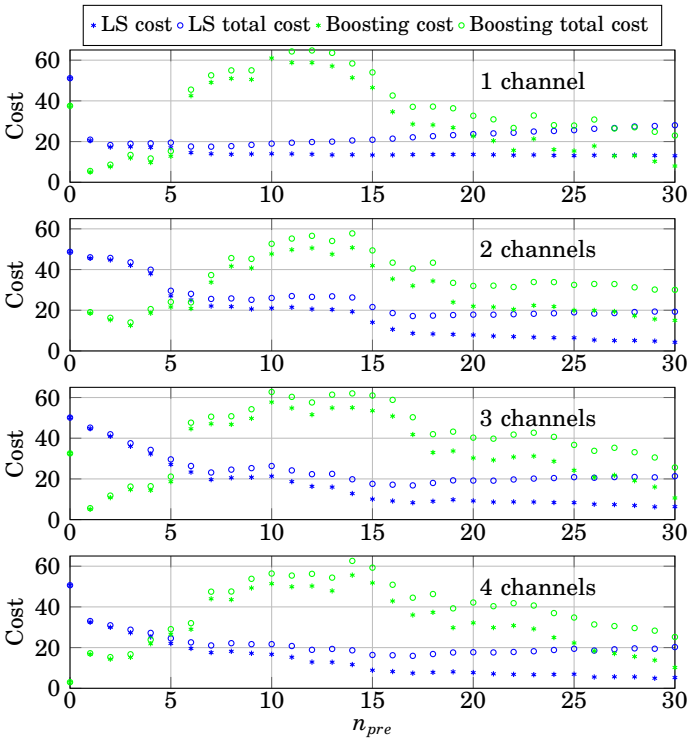


Figure 5.5 The result when the number of samples before the transient, n_{pre}, was varied. The cost refers to the cost function (5.1) with $cost_{FP} = cost_{FN} = 1$. The total cost also contains the extra penalty term for large n_{pre}.

parameter from 0 to 30, with the other parameters decided by the previous experiments, is displayed in Fig. 5.5. The boosting classifier still has some problems in the middle region, while the LS classifier shows a monotone-like performance increase. The performance measure constant p_2 was chosen as 0.5, such that it gave a slowly increasing cost for growing n_{pre}. The final parameter choices are summarized in the left part of Table 5.1.

The considered scenario is in some sense asymmetric, as was concluded in Sec. 5.2. To see the effect of including the asymmetry, the entire training procedure was performed one more time, with $cost_{FP}$ being ten times as large as $cost_{FN}$. The final parameter choices are listed in the right part of Table 5.1. It can be seen that the parameters for the initial training with LS are similar to the symmetric case, but that n_{post} is much larger

Table 5.1 Result of initial training. The left part shows the symmetric case and the right part the asymmetric case.

Initial training		$cost_{FP} = cost_{FN}$				$cost_{FP} = 10cost_{FN}$			
Least-squares	Channels	$[\psi]$	$[x\ \theta]$	$[x\ \phi\ \theta]$	$[x\ y\ \phi\ \theta]$	$[\psi]$	$[y\ \psi]$	$[y\ \theta\ \psi]$	$[x\ y\ \theta\ \psi]$
	n_{pre}	7	17	17	17	3	5	7	7
	n_{post}	2	2	2	2	1	2	3	3
Boosting	Channels	$[\psi]$	$[\theta\ \psi]$	$[x\ \phi\ \psi]$	$[x\ z\ \phi\ \psi]$	$[\psi]$	$[z\ \psi]$	$[y\ z\ \psi]$	$[y\ z\ \phi\ \psi]$
	n_{pre}	1	3	1	0	8	4	5	8
	n_{post}	1	0	1	0	15	15	16	15
Least-squares and boosting	Channels	$[\psi]$	$[x\ \theta]$	$[x\ z\ \psi]$	$[x\ y\ \theta\ \psi]$	$[\psi]$	$[y\ \psi]$	$[y\ \theta\ \psi]$	$[x\ y\ \theta\ \psi]$
	n_{pre}	1	17	2	1	3	9	4	2
	n_{post}	1	2	2	1	14	14	14	14

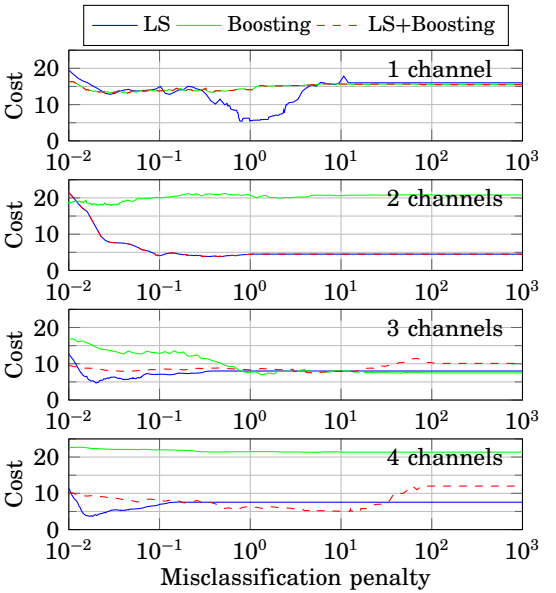


Figure 5.6 The classification results for training the primal SVM classifier. The cost refers to the cost function (5.1) with $cost_{FP} = cost_{FN} = 1$.

for the other initial training methods. The difference is that the LS classifier was very careful and made almost no false positive classifications, but the other two initial training methods had some false positives for low values of n_{post}. Only a few false positives gave a large cost, so large that not even the quadratic penalty could force a low value of n_{post}. For the initial training with LS, the quadratic penalty was the dominating term as there were mostly false negatives, and the result was a low value for n_{post}.

SVM training Parameters for the SVM classifiers were found by gridding the parameter space, as described in Sec. 5.2. For the primal SVM classifier, with equal cost for false positives and negatives, the result of the parameter search is displayed in Fig. 5.6, which shows the cost (5.1) with $cost_{FP} = cost_{FN} = 1$ for different values of the misclassification penalty. It can be seen that no parameter setting gives perfect classification. The best result was achieved with four channels used and the initial training made with LS. All the best SVM classifiers for the different initial training methods are listed in Table 5.2.

An example result from the grid search for the parameters for the dual SVM classifier is displayed in Fig. 5.7. The asymmetric approach was used

Table 5.2 Result of SVM training. The left part shows the symmetric case and the right part the asymmetric case.

		$cost_{FP} = cost_{FN}$				$cost_{FP} = 10cost_{FN}$			
Initial training		n_{ch}	P	l	Average cost	n_{ch}	P	l	Average cost
Least-squares	Primal SVM	4	0.0182	-	3.65	2	0.571	-	10.60
	Dual SVM	2	6.91	67.17	1.80	2	320.75	668.57	3.0
Boosting	Primal SVM	3	1.33	-	7.0	2	0.00750	-	5.80
	Dual SVM	4	7.97	7.97	2.80	2	28.63	537.00	0.0
Least-squares and boosting	Primal SVM	2	0.294	-	3.85	4	0.0330	-	3.20
	Dual SVM	2	6.91	67.17	1.80	1	33.00	320.75	0.0

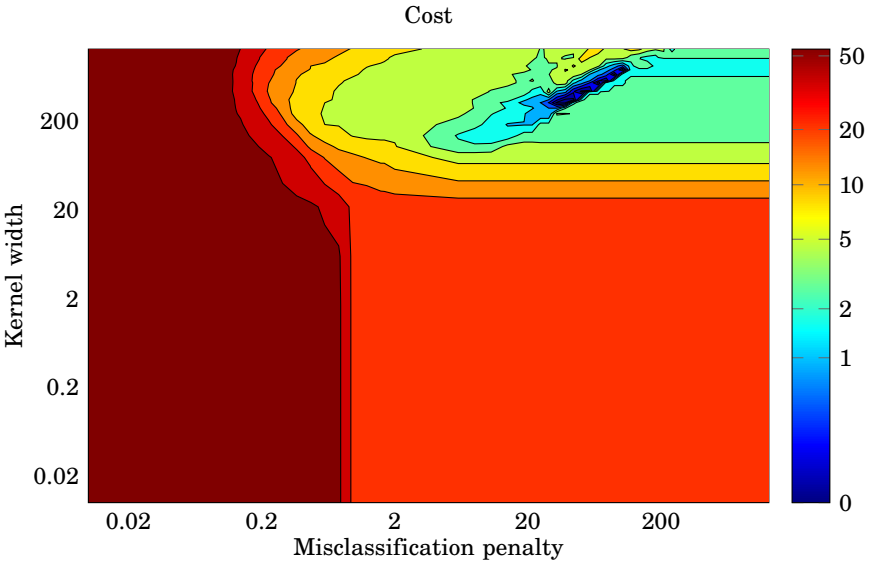


Figure 5.7 The classification results for training the dual SVM classifier using the asymmetric approach with one channel and where the combination of LS and boosting was used for the initial training. The cost refers to the cost function (5.1) with $cost_{FP} = 10cost_{FN}$.

with one channel, and the initial training method was the combination of LS and boosting. Although hard to see, some of the parameter settings give perfect classification, and one of them is listed in Table 5.2.

The best classifiers for all choices of number of channels and initial training are listed in Table 5.2. It can be seen that the best performance was achieved with the asymmetric approach, giving perfect classification. The main difference between the two approaches was the choice of n_{post} versus n_{pre}. In the symmetric case, n_{post} was small and n_{pre} usually larger, resulting in a classification with only a small delay. In the asymmetric case, however, n_{post} was large and n_{pre} small when boosting and the combination of LS and boosting were used for the initial training, giving a longer delay, but on the other hand giving perfect classification. Depending on sampling time, this delay might be of more or less importance. In this scenario, with a sampling time of 4 ms, the extra delay in the asymmetric case becomes 40–50 ms, which is quite short and this suggests that the penalty for choosing n_{post} might have been a bit too large.

The results from the SVM training give no unanimous answer to how many channels that should be used, as there were classifiers performing

the best in all cases, with one to four channels used. A feasible strategy could be the one taken in this chapter, i.e., evaluate a number of different choices of number of channels and choose the one performing best. The best option for the initial training was the combination of LS and boosting, giving the best performance in all cases except for the primal SVM classifier in the symmetric case.

The reduction in assembly time for using the transient detection compared to the threshold detector earlier used was approximately 0.31 s, with a standard deviation of 0.064 s.

Slide motion transient

The training procedure was applied also to the slide motion transient, see the recorded transients in Fig. 5.8. Based on the results from the snap detection, the combination of LS and boosting was used for the initial training. The same parameter values for p_1 and p_2 were used. For this transient, both the symmetric and the asymmetric case gave low values for n_{post} and somewhat larger for n_{pre}. The final SVM training resulted in perfect classification for both the primal and the dual variants. This is an indication of that this transient was easier to detect than the snap.

The reduction in assembly time for using the transient detection compared to the threshold detector earlier used was approximately 0.18 s, with a standard deviation of 0.041 s.

Real-time implementation

Classifiers for the two transients were implemented and used for executing the assembly operation. The classifiers chosen for implementation were both dual SVM classifiers that resulted in perfect classification for training and validation data. The asymmetric formulation was used, and the previously used thresholds were used as backup classifiers. The number of support vectors was 35 for the slide motion classifier and 53 for the snap motion classifier. As was mentioned in Sec. 5.2, all available switches were used for training/validation, so the experiments were performed using a subset of these switches, namely two switches from the training data set and two from the validation data set.

An unforeseen problem occurred when the classifiers were both used in the assembly operation. The shape of the force/torque signals directly following the slide motion transient were similar to the snap motion transient, cf. Figs. 5.1 and 5.8. These data were not used for training the snap classifier, and the classification algorithm was therefore confused when faced to these data and the result was quite many false positive classifications right in the beginning of the snap motion. One solution to this problem would be to include these data for training. Another solution was

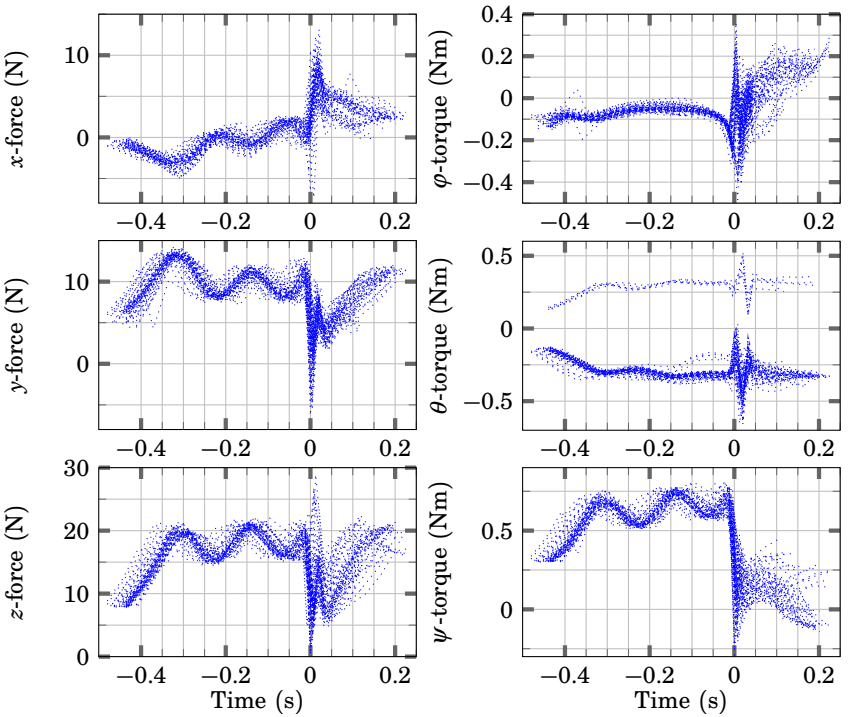


Figure 5.8 The recorded data from the slide motion. To increase the readability, all sequences have been shifted such that the transient occurs at $t = 0$ s. The force/torque directions refer to the feature coordinates defined in Sec. 3.2.

to wait for the slide motion transient to die away before the snap classifier was activated. In Fig. 5.8 it can be seen that the slide motion transient has died away after around 0.1 s, and in Fig. 5.1 it can be seen that it takes at least 0.3 s from the snap motion is started until the snap occurs. The time to wait was therefore set to 0.1 s.

The assembly operation was performed 20 times. None of the transients were missed and no false positive classifications were made. The total reduction in assembly time by using the classifiers was in average 0.49 s (standard deviation 0.070 s), which meant a reduction of the total time to insert the switch in the bottom box with approximately 15 %.

5.4 Discussion

The classifiers considered in this chapter were supposed to replace simple threshold detectors that have been used before. The automatic procedure for training the classifiers makes it possible for inexperienced robot operators to get robust classifiers in a simple way. The advanced classifier can also be used to for enable detection of contact events with a lower force level, as compared to the threshold detector. The two investigated state transitions showed that there also is time to gain, i.e., the production can be sped up.

The procedure described in Sec. 5.2 to pick out background sequences to include into the training data was applied as using all background sequences for training would be too computationally expensive. On the other hand, it was computationally cheap to perform validation on all the background sequences. That made it possible to start with a small set of background sequences, and include more if needed, as this will lead to a more robust classifier.

One of the design goals of the presented procedure was to reduce the number of channels used. This was done by investigating all possible combinations of channels and in the end use the one giving the best result. An alternative could be to employ Principal Component Analysis (PCA), and instead find one or more linear combinations of channels to use. PCA would pick out combinations of channels with large variance, but this might not be the important information for doing classification. It remains as future work to investigate this approach.

The complexity of the classifiers has been kept low, in the sense of using as little data as possible. Aside from keeping the training time from increasing and also making classification faster, it will give some robustness. In an assembly scenario like the one considered in this chapter, the training data size will be quite small, at least initially. A too complex classifier will then tend to model also the noise, and therefore being a worse classifier when faced to new data. The parameters used for choosing the complexity of the classifiers (p_1 and p_2) in the example scenario may, however, have given too much emphasis on low complexity. This was especially the case when boosting was used as the initial classifier for the snap, when very few samples before and after the snap were used for the classifier.

In a production setting, all erroneous classifications should be saved and, perhaps, also the successful ones. These data should be used to redo the training of the classifier to improve performance. With more data available, maybe also the complexity of the classifier can be increased.

Automating a task like the one considered in this chapter is difficult. Every new task will be different, with different requirements. Still, a

systematic approach, such as the one presented, will be a step towards a complete solution. The experiments performed show a proof of concept that it works.

Performing the training procedure takes a really long time, which may take in the order of days on a standard computer, depending on the grid size used in the SVM training. But the training procedure is possible to parallelize and thus to use multiple machines. It would therefore be perfect as a cloud service. Record some training data, send them to the cloud, and receive a classifier in response. With large computational resources it might further be feasible to vary more parameters, e.g., other kernels for the dual SVM classifier, as well as other types of classifiers.

5.5 Conclusions

A systematic procedure for finding machine-learning based classifiers to detect transients in force/torque data was presented, but it would also be feasible to use the method with any other time-series data. A force-controlled assembly task was used as an experimental case to show that the method worked. In the implementation of the assembly task, the classifiers were used to replace simple threshold detectors that were used before. The replacement resulted in the total assembly time being reduced with 15 %.

6

Generation of Sensor-Based Robot Programs from High-Level Task Specifications

6.1 Introduction

When a human teaches another human a task, a high-level description is most commonly given. A simple assembly task, such as the snapfit assembly task in Chap. 3, can be taught by describing only the final configuration, i.e., the geometric relation between the switch and the bottom box. Details, such as how to grasp the parts and what forces to apply, can usually be learned by experimentation, and they do not need to be part of the task instruction. To deploy the same assembly task on an industrial robot involves a substantial amount of programming work, e.g., specifying grasping positions and trajectories for the robot. Extensive background knowledge and experience of the application domain are usually required. Even simple assembly tasks may demand many days of implementation work for skilled system integrators.

Programming standard position-based robot programs can be performed using so called engineering tools, such as RobotStudio for ABB robots [ABB Robotics, 2015c], see an example of a YuMi robot station in Fig. 6.1. A graphical user interface together with a simulation environment makes it quite straightforward to program a task. The robot can be taught with a high-level description, i.e., specifying certain positions to go to, and the motion between these points is then generated automatically for each joint. These engineering tools, however, do not offer any functionality to incorporate external sensors, e.g., a force sensor, in the task

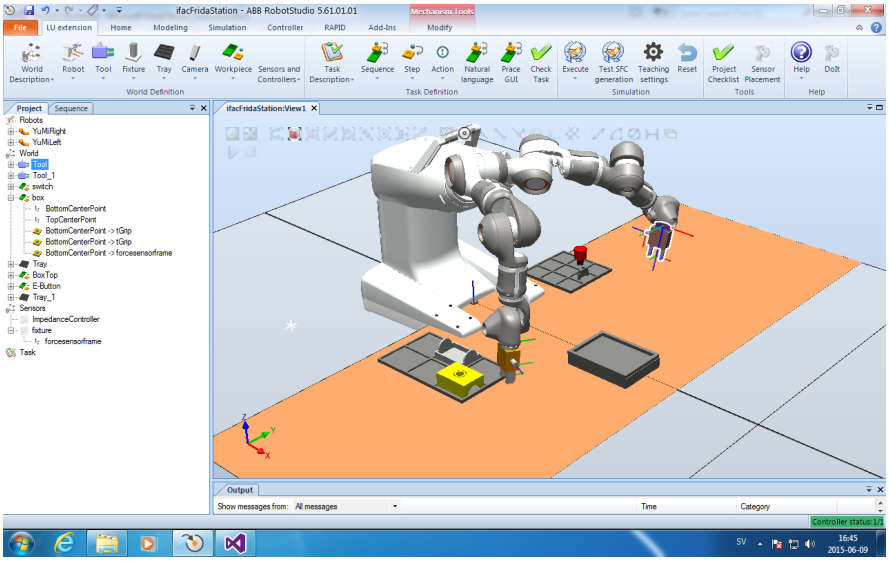


Figure 6.1 A screenshot from the engineering tool. The robot station is the one that is used for the experiments in this chapter.

programming. The work presented in this chapter is aimed at extending the engineering tool with possibilities to program also sensor-based skills on a high level. This can be done by either loading ready-to-use skills from a library, or to create the skills from scratch. The high-level task description from the engineering tool is translated to a low-level description that is executable on the robot system.

A widely used concept for programming industrial robots is online programming, i.e., using the teach pendant, a control unit with joystick, to manually move the robot between positions to be used in the program. In [Pan et al., 2012] a survey of different methods for programming industrial robots was presented. It is usually desirable to minimize the downtime for the robot, and offline programming methods are therefore more suitable, where the programs can be simulated before the actual robot is needed, e.g., the approach presented in [Mitsi et al., 2005]. Visual programming languages and graphical user interfaces can be used to simplify the programming for non-expert operators. The available tools, however, only offer generation of programs using native code of the robot controllers, usually only including standard position-based control.

An approach to high-level programming of sensor-based tasks that is close to the one taken in this chapter is presented in [Beetz et al., 2010], where high-level actions are translated into robot programs, using

knowledge-based techniques. The system is designed for mobile robots and the resulting code is at the level of ROS primitives [ROS, 2015], which does not provide any real-time guarantees often needed in an industrial setting. The same research group also presents an approach to map high-level constraints to control parameters in order to flip a pancake with a robot [Kresse and Beetz, 2012]. A convenient way of hiding the detailed control structures is using reusable skills or manipulation primitives [Kröger et al., 2011]. They become an interface to the low-level control, and they are similar to the iTaSC specified motions used in this thesis. The manipulation primitives are more involved than the iTaSC motions, including such functionality as stop conditions, tool commands, error handling, etc., and thus require more implemented low-level code than is the case for the method in this chapter.

Using skills as building blocks to program robotic tasks is another approach similar to the method in this chapter. In [Pedersen et al., 2014] skills are combined in a simple way using a touchpad, and gestures are used for teaching parameters. The approach does not, however, offer any simple way to create new skills, and it is focused on simple skills such as pick-and-place. A similar method is also described in [Waibel et al., 2011], where predefined skills are combined into action recipes. The focus is, however, not on the specification of tasks, but on reusability of the task specification and sharing information between robots. Another method for reusability and task transfer between robot systems is described in [Huckaby and Christensen, 2012], where a taxonomy for modeling assembly tasks using skill primitives is presented.

6.2 High-level task specification

When an assembly task is specified, the highest level is the assembly graph. It is represented by a partially ordered tree of assembly operations. An example for the emergency stop button use case was given in Fig. 3.6, and the corresponding assembly graph as visualized in the engineering tool is displayed in Fig. 6.2. The leaves correspond to the parts to be assembled, and the root node to the resulting assembled product. The intermediate nodes between the leaves and the root correspond to sub-assemblies. Each assembly operation is specified by the desired geometrical relation between the parts involved. These relations are specified as constraints between coordinate frames that are attached to the involved objects. Such a constraint can be that the frames on two different objects should coincide.

Task specification on a high level is suitable to perform in a simulation environment. Coordinate frames attached to different objects can

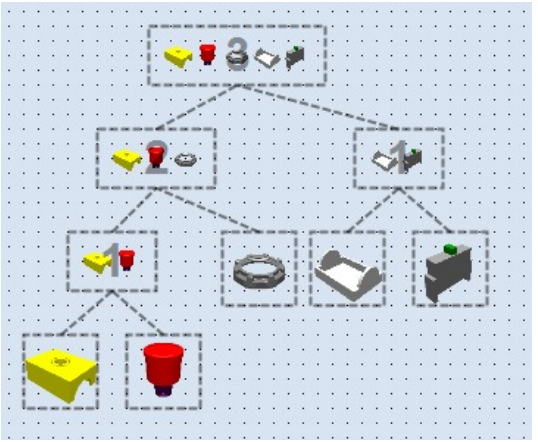


Figure 6.2 The assembly graph for the emergency stop button assembly scenario as visualized in the engineering tool. Note that the boxes with multiple components represents the assembled (sub)products.

then easily be specified and visualized, see an example in Fig. 6.3. Constraints can also easily be specified, for instance, by moving the objects in the simulation environment to the desired configuration. An example is displayed in Fig. 6.4, where the relation between the fixture and the bottom box in the snapfit assembly scenario (Sec. 3.3) is specified.

To make it possible for a robot to accomplish an assembly task, such things as which type of gripper to use and how to grasp the objects must also be specified. It would be good functionality if the program could automatically do this matching based on the available grippers and the geometry of the parts. This is an active area of research, and future work includes integrating existing algorithms. It is, however, also quite straightforward for the user to manually specify this information.

A step towards an executable robot program from the assembly graph is to create sequences of operations. These sequences should take such things as the number of robot arms, and the grippers and sensors available into consideration when creating a program. For instance, if the snapfit assembly should be performed using one manipulator arm together with a fixture, the following sequence could be the solution:

- Pick the bottom box
- Put the bottom box in the fixture
- Pick the switch
- Perform the snapfit assembly

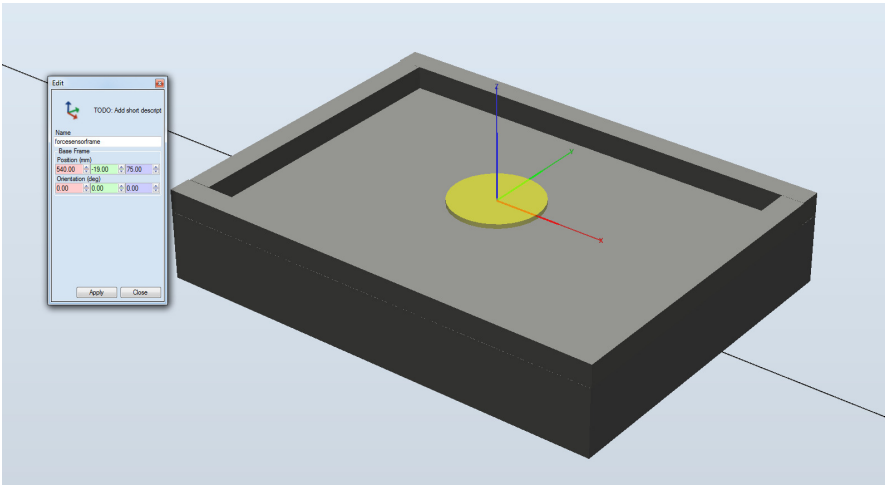


Figure 6.3 A screenshot from the engineering tool, which shows how the coordinate frame on the fixture is visualized. The frame-axis coloring follows the rgb-convention, that is, the red axis is the x-axis, the green the y-axis, and the blue the z-axis.

- Move the final assembly away from the fixture

Each of the operations in the sequence can either be performed using standard position-based control, or be sensor-controlled skills. If the parts are placed in well-defined trays, then the robot can pick the parts using position-based operations, otherwise a vision system might be needed for the system to be able to localize the parts. The assembly operations are, on the other hand, more suitable to perform using force control, in order to be able to handle position uncertainties and part variations.

In a high-level programming environment, the system should be able to automatically generate these sequences for each node in the assembly graph. Based on the available devices, such as the type of robot, grippers, sensors, etc., the system should know if position-based operations are possible to perform, or if sensor-based skills are necessary. In case a sensor-based skill is needed, the system should be able to choose a suitable skill from the library, or if no appropriate skill was available, prompt the user to create a new skill from scratch. In the implementation, a sequence was automatically created from the assembly graph, but which skill to use had to be chosen manually. The user could choose where an assembly operation should be performed, i.e., in which fixture, and the system then automatically generated the code for picking the components and for the movements to the fixture.

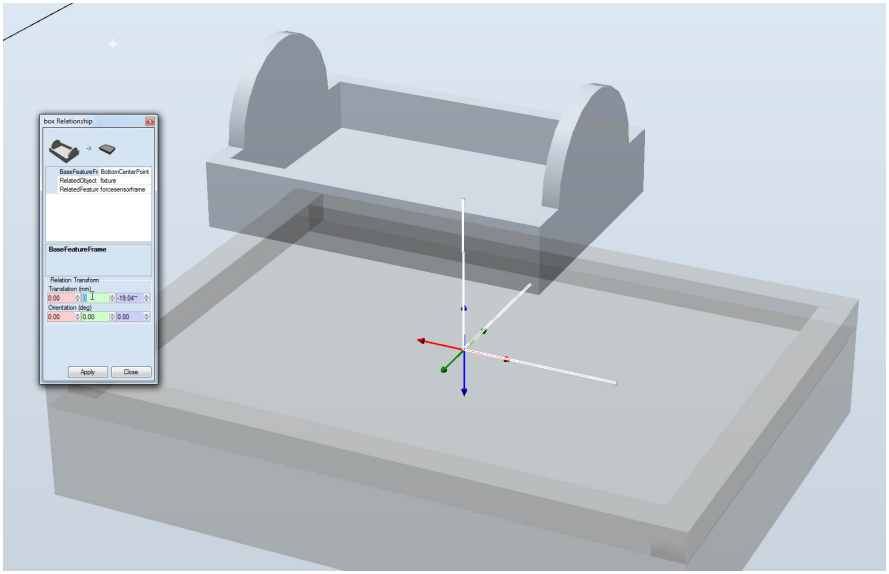


Figure 6.4 Specification of a geometric relation between the fixture and the box in the engineering tool.

Another even more high-level functionality is natural language support, such that the user can express a task using natural language sentences. The sentences are parsed and semantically analyzed in order to identify the operations to be performed and objects to be manipulated. In [Stenmark and Malec, 2014] an approach using natural language for specifying assembly tasks is presented.

Generation of position-based robot programs is already today available for commercial industrial robot systems. The novelty is to add sensor-based skills, including the whole chain from a high-level description to low-level executable code. This functionality was integrated with the existing one in the engineering tool. The sensor-based skills can either be predefined skills that are saved in a library, or be created in the system. The skills in the library should have a description saying what they do, what they require in terms of devices (robots, sensors, fixtures, etc.), and what parameters they require to be set. The skills to be created can be in the form of guarded search motions, i.e., a search motion in a direction that is ended when the corresponding contact force is detected. Most assembly tasks can be accomplished by using a number of guarded motions, including all assembly use-cases that are considered in this thesis.

A guarded motion is defined by a search direction, a search velocity, and a force/torque threshold level that should be exceeded to finish the

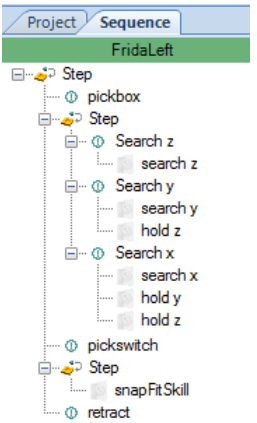


Figure 6.5 The sequence of operations for the snapfit example, as shown in the engineering tool.

motion. In an assembly task, it can be useful to maintain established contacts by using force control, e.g., if a search should be performed along a surface and the search direction does not match the surface, then either contact will be lost or the contact force will increase. Keeping contacts using force control can also handle searches along surfaces that are not flat. The guarded motion may be specified to perform force control in certain directions by specifying force constraints, i.e., to keep the desired force in a direction. Each force constraint must be specified with desired force value, direction and a force controller with parameters.

Example snapfit

The snapfit assembly scenario, described in Secs. 3.2–3.3, will be used as an example to illustrate how an assembly task can be specified. The left arm of YuMi was used, and to accomplish the assembly a fixture placed in front of the robot was used. A force/torque sensor was mounted beneath the fixture, such that force control could be used when carrying out operations in contact with the fixture. An overview of the robot station in the engineering tool is displayed in Fig. 6.1. The sequence of operations, as shown in the engineering tool, is displayed in Fig. 6.5. Firstly, the bottom box had to be picked from its tray, which could be performed using a position-based operation called `pickbox`. This was a standard operation, which contained the following actions

- Go to a position above the tray
- Move linearly down to the gripping position

- Close the gripper
- Move linearly up above the tray

The next operation was to put the bottom box in the fixture. Performing this operation using position-based control was not a good alternative as the margins in the fixture were small. Further, the tray was slightly larger than the box, which resulted in uncertainty of the position of the grasped box in the gripper and thus made it even more difficult to fit the box into the fixture using position-control. A force-controlled strategy was therefore used. The robot started in a position above the fixture, and then three consecutive guarded motions were used to push the box into a corner of the fixture. The coordinate frame used is displayed in Fig. 6.3. The first guarded motion was performed along the negative z-direction. The next search was along the negative y-direction. During this search a constraint was added to keep the z-force, i.e., the box kept the contact with the fixture and eventually slid down into it. Finally, a guarded search in the negative x-direction was added, with constraints to keep the forces in the y- and z-directions.

The following operation was to pick the switch from its tray, called `pickswitch`, which could be performed using a similar operation as the picking of the bottom box. The next operation was to attach the switch to the bottom box. This was performed by reusing a previously defined skill, the `snapFitSkill`. It took a number of parameters, such as search velocities and tool transformation, which could be set by the user. The last step was to move the final assembly away from the fixture, using an operation called `retract`.

6.3 Low-level task specification

The interface to the robot was assumed to be the one described in Chap. 2. The low-level task specification was therefore about generating motions for each joint of the robot. The iTaSC-framework, as described in Sec. 3.2, was used for doing this. To be able to do it, at least one kinematic chain was needed, together with a number of outputs to control. A guarded motion is an operation with a constant velocity in the search direction, i.e., the corresponding output should be velocity-controlled. The force constraints correspond to outputs being force-controlled. To completely specify the task, i.e., all six degrees of freedom, any remaining Cartesian directions were position-controlled to keep their positions.

The state machine coordinating the simple search motions was also a part of the low-level task specification. Each state corresponded to one

guarded search motion and the transition conditions were based on force measurements using threshold levels, as was described in Sec. 3.2.

6.4 Code generation from high-level to low-level task specification

Position-controlled motions could be executed by the native robot controller by using the available move commands. For sensor-controlled motions, however, an external controller had to be used, and a coordinating skill state-machine had to be generated.

The description of the sequence of guarded search motions created in the engineering tool was extracted into an intermediate xml-format. An example of what this format looks like is given below, which shows the skill from Fig. 6.5, i.e., to put the bottom box in the fixture. First a number of initial declarations are given, such as coordinate frames, tool transformations, and force controller parameters. After the initial declarations, a number of actions follow. The actions are to be executed in the order they are appearing in the xml-document. Each action should have one `Direction`, which defines in which direction the guarded search motion is to be performed. The search speed and the force threshold level should also be specified. The actions can also have constraints, i.e., directions that are force controlled. The constraints must specify which controller to use and a force reference value, together with the direction that should be force controlled.

```
<?xml version="1.0"?>
<SkillSpecification>
       <Frame id="f1">
              <origin>[ 490 , 6 , 43 ]</origin>
              <quaternion>[ 1 , 0 , 0 , 0]</quaternion>
       </Frame>

       <ToolTransform id="tool1">
              <trans>[0,0,87]</trans>
              <quaternion>[0,-0.707106781,0.707106781,0]</quaternion>
       </ToolTransform>

       <ImpedanceControlParams id="z-controller">
              <M>0.01</M>
              <D>0.2</D>
       </ImpedanceControlParams>

       <ImpedanceControlParams id="y-controller">
              <M>0.02</M>
              <D>0.6</D>
       </ImpedanceControlParams>
```

```
        <Action id="z-search" tool="tool1">
                <Direction>
                        <searchVelocity unit="mm/s">-30</searchVelocity>
                        <motionframe>f1</motionframe>
                        <motiondir>z</motiondir>
                        <threshold unit="N">3</threshold>
                </Direction>
        </Action>
        <Action id="y-search" tool="tool1">
                <Direction>
                        <searchVelocity unit="mm/s">-40</searchVelocity>
                        <motionframe>f1</motionframe>
                        <motiondir>y</motiondir>
                        <threshold unit="N">3</threshold>
                </Direction>
                <Constraint>
                        <type>forcecontrolled</type>
                        <controllerId>z-controller</controllerId>
                        <motionframe>f1</motionframe>
                        <motiondir>z</motiondir>
                        <value unit="N">3</value>
                </Constraint>
        </Action>
        <Action id="x-search" tool="tool1">
                <Direction>
                        <searchVelocity unit="mm/s">40</searchVelocity>
                        <motionframe>f1</motionframe>
                        <motiondir>x</motiondir>
                        <threshold unit="N">-6</threshold>
                </Direction>
                <Constraint>
                        <type>forcecontrolled</type>
                        <controllerId>z-controller</controllerId>
                        <motionframe>f1</motionframe>
                        <motiondir>z</motiondir>
                        <value unit="N">2</value>
                </Constraint>
                <Constraint>
                        <type>forcecontrolled</type>
                        <controllerId>y-controller</controllerId>
                        <motionframe>f1</motionframe>
                        <motiondir>y</motiondir>
                        <value unit="N">3</value>
                </Constraint>
        </Action>
</SkillSpecification>
```

The program used for executing the state machines, JGrafchart, takes state machine descriptions also in an xml-format. The JGrafchart specification also includes information about how the state machine should be displayed in the graphical user interface. The JGrafchart specification had to be generated by the system, based on the skill specification xml-document, such as the one given above. The information about the

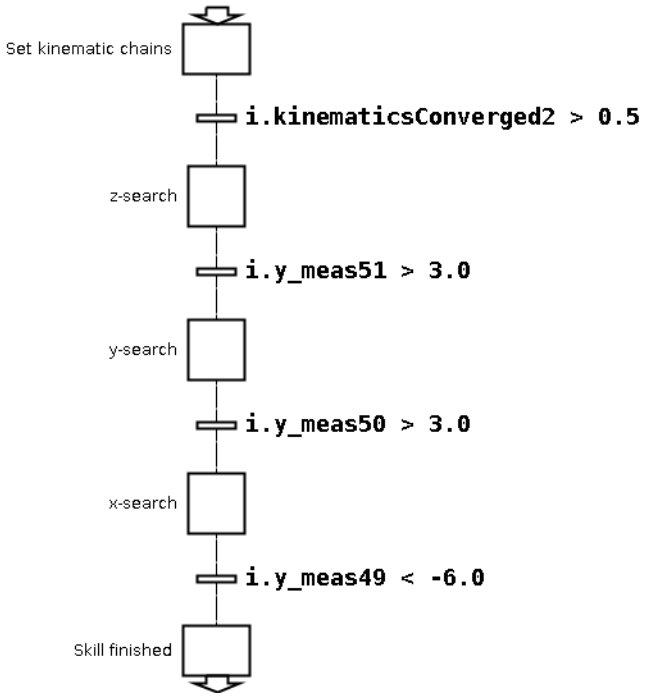


Figure 6.6 The generated state machine as displayed in JGrafchart. The signals beginning with i.y_meas refer to measurement signals from the external controller, where numbers 49–51 correspond to the forces in the x-, y-, and z-directions, respectively.

skill also had to be transformed to fit the interface to the external controller. For instance, a kinematic chain must be specified. One kinematic chain was created for each coordinate frame defined in the xml-document. The chain was defined in such a way that the feature coordinates corresponded to the directions of the specified coordinate frames. Procedures for initialization of an external skill, including the handover from the native controller to the external controller, also had to be added, as well as procedures for when a skill is finished, i.e., the handover back to the native controller. The generation of the executable state machine, i.e., the JGrafchart specification xml-file, was handled by a program that could be called as a service from the engineering tool. The resulting state machine could be opened and examined in JGrafchart, see the state machine in Fig. 6.6, and it could also be edited. Modifications of the state machine could, however, not be propagated back to the skill description xml-document and the engineering tool.

The execution of a complete task was initiated from the engineering tool, which called the program for generating the executable state machine. The execution of the task was coordinated from JGrafchart, which either ran sensor-based skills or native robot instructions, which were commands sent as strings via a network connection to a program running on the native controller.

6.5 Experimental results

In order to verify that the code generation worked as expected, the sequence displayed in Fig. 6.5 was used as a test case. The generated code was executed on the robot system several times with successful task completion each time. Sensor-based skills have further been generated also for the IRB120 and IRB140 robots.

6.6 Discussion

The presented system makes it possible for non-expert users to create sensor-based skills from scratch. No low-level knowledge, such as how to implement control algorithms and how to use the iTaSC framework, is required. Kinematic chains, tool transformations, and other parameters are extracted from the engineering tool and code generation is used to transform it all into a low-level description. The system further makes it possible to combine predefined skills into advanced tasks. All these features significantly simplifies the task specification for the user. The system is, however, not yet especially intelligent, in the sense that most of the task description must be created manually. Making the procedure more automatic would simplify the task specification even more for the user, and it is something that is part of the future work. Another simplification for the user would be if demonstrations can be used for specifying sensor-based skills, i.e., extracting search directions, force levels, and force constraints.

The current implementation has quite some limitations. The external control program is fixed, e.g., with hard-coded robot kinematics and sensor calibration. To make the system more flexible, also the external controller should be automatically generated from the specification such that both robots and sensors can be modified. This would also enable easy deployment on a new robot system, as no dependency of any existing implemented code then is needed.

The system enables non-expert users to create sensor-based skills from scratch. When such a skill is used for performing assembly, there will most likely occur unexpected things, which should be taken care of by adding

error detection and recovery. This should, however, be performed in a skill editor, where the state machine is visible. As yet, modifications made to the state machine were not propagated back to the engineering tool. Extra states for error handling might be hard to visualize in the sequence format as displayed in Fig. 6.5. Error handling might be something that should be handled by expert users and, then, it should be appropriate to use an external tool, e,g., JGrafchart that was used in this work.

The guarded search motions are quite simple. One possible extension would be to add multiple conditions to finish the motion instead of only a force threshold. For instance, a timeout condition could be used to handle cases where the surface that is searched for is not where it was supposed to be. The force constraints require force controllers with parameters. Tuning of these parameters is usually tedious work, and an automatic tuning strategy, such as the one presented in Chap. 7, would be very useful.

6.7 Conclusions

It was presented how an engineering tool was extended such that it was possible to program sensor-based skills on a high level. Multiple levels of task specification were considered, where the specification was refined at each level. Code generation was used to transform the high-level specification into an executable program on the robot system.

7

Adaptation of Force-Control Parameters

7.1 Introduction

There is a need to make it easy for robot operators to specify tasks, especially when external sensing is used. One such example is force-controlled assembly. Force sensing is beneficial in these tasks, as it increases the robustness towards uncertainties, e.g., caused by inaccurate gripping, compared to for instance a position-controlled implementation. The environment is often stiff, which makes it crucial to design appropriate force controllers to maintain stability of the force-controlled system. This is a non-trivial task that may be hard for the task programmer. One solution to this problem is to offer a self-tuning mechanism, making the force controllers adaptive.

In this chapter, the problem of robotic assembly based on force sensing only is addressed. An adaptive algorithm for choosing force control parameters in a predefined controller structure is presented. A contact model is identified, and it is used to tune the force controller. The approach is finally integrated in the snapfit assembly scenario (Chap. 3).

Identification of contact model parameters has previously been considered by many researchers. In [Erickson et al., 2003], four different methods for estimating the environment contact model were described and experimentally verified. One of the methods was originally presented in [Love and Book, 1995], which describes how the parameters in an impedance controller can be chosen when using contact model parameters estimated with the Recursive Least Squares (RLS) method. A comparison of different algorithms for real-time identification of contact model parameters are described in [Haddadi and Hashtrudi-Zaad, 2008], among them RLS. In [Roy and Whitcomb, 2002], an adaptive force controller was presented, being based on an estimate of the contact stiffness. A similar

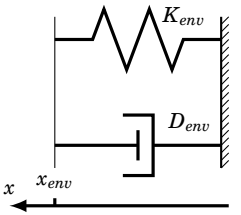


Figure 7.1 Contact model.

approach was presented in [Kröger et al., 2004], which considers adaptive force controllers within the Task Frame Formalism. An adaptive force controller extending a position controller is presented in [Villani et al., 1999]. The position reference is scaled in the force-controlled direction such that the force can be controlled. The scaling factor is made adaptive when the stiffness of the environment is unknown. In [Mallapragada et al., 2006] estimates of contact stiffness and damping are used in a gain scheduler based on an artificial neural network for a PI force controller.

An approach to identification of a contact model with multiple contact points is given in [Weber et al., 2006]. The geometry was assumed to be known and this made it possible to calculate the contact locations; the results presented are based on simulations. An extension of the results provided in [Weber et al., 2006] is [Verscheure et al., 2010], which also considers geometric uncertainties and presents experimental results.

A method for designing force controllers when given environment stiffness by the robot user was presented in [Natale et al., 2000]. An industrial robot with a position-controlled interface is assumed, and the robot dynamics are taken into consideration when doing the controller design.

7.2 Modeling

Contact model

The environment is modeled to consist of a spring and a damper, according to Fig. 7.1, where x denotes a Cartesian position coordinate. Thus, interacting with the environment gives the reaction force, F, given by

$$F = K_{env}(x_{env} - x) - D_{env}\dot{x} \tag{7.1}$$

where $\dot{x}$ denotes the velocity in the direction x. The stiffness of the environment is denoted by K_{env}, the damping by D_{env}, and the location of the unloaded environment by x_{env}. The environment is further assumed to be decoupled, such that there is one relation (7.1) for each Cartesian direction.

The environment perceived by the robot will not equal the actual environment, it will rather be a combination of the stiffness and damping properties of the tool attached to the robot (the force sensor), the robot itself, and the actual contact. Hence, a stiff environment might be perceived as a soft one if the tool on the robot is soft. Further on, if the tool has different stiffness properties in different directions, even an isotropic contact material will be perceived to have different stiffness in different directions.

Adaptation algorithm

The algorithm chosen was the Recursive Least Squares (RLS) method [Johansson, 1993]. The contact model (7.1) is nonlinear, because of the product $K_{env}x_{env}$. This product can, however, be seen as a separate model parameter, and then the model is linear. It can be cast in regressor form according to

$$y = \varphi^T\theta \quad , \quad \begin{cases} y = F \\ \varphi = \begin{bmatrix} -x & -\dot{x} & 1 \end{bmatrix}^T \\ \theta = \begin{bmatrix} K_{env} & D_{env} & K_{env}x_{env} \end{bmatrix}^T \end{cases} \tag{7.2}$$

where all parameters have been gathered in θ.

The RLS algorithm is given as [Johansson, 1993]

$$\begin{cases} \hat{\theta}_k = \hat{\theta}_{k-1} + P_k\varphi_k\varepsilon_k \\ \varepsilon_k = y_k - \varphi_k^T\hat{\theta}_{k-1} \\ P_k = \frac{1}{\lambda}\left(P_{k-1} - \frac{P_{k-1}\varphi_k\varphi_k^T P_{k-1}}{\lambda+\varphi_k^T P_{k-1}\varphi_k}\right) \end{cases} \tag{7.3}$$

where the subindex k denotes the sample index, and y, φ, and θ are defined in (7.2). The update of the parameter estimate, $\hat{\theta}$, is calculated as the previous estimate plus a correction term based on the estimation error ε. The adaptation gain matrix P is propagated according to the last equation. The *forgetting factor* λ can be used to cope with time varying parameters by setting it to a value less than 1. A value of $\lambda = 1$ gives the usual least-squares solution. An initial value for the parameters and the adaptation gain matrix have to be chosen. Usually P is chosen to have a large magnitude, which means that the initial guess of the parameters is not to be trusted. The large P will further lead to a fast convergence of the parameters.

Each force controlled direction will have nominal controller parameters, likely not well tuned. These will be used during the phase when the contact model parameters are estimated. To assure that the input signals to the estimator are persistently exciting [Johansson, 1993], the force reference is set to a sufficiently exciting signal, e.g., a square wave. As the

covariance of the estimate is decreased (proportional to the P-matrix), the controller parameters are updated based on the contact model parameter estimates. Once the covariance is considered to be low enough, this adaptation phase is finished.

Force controller

The force controller used in the assembly framework was decoupled impedance control [Hogan, 1985] for each Cartesian direction x. This setup makes it possible to perform force control in some directions and, e.g., position control in others. The control law used for the impedance controller is given by (7.4).

$$\ddot{x}_{des} = \frac{1}{M}\left(F_x - F_{x,ref} - D\dot{x}_{des}\right) \tag{7.4}$$

The controlled direction is denoted by x and its desired behavior by x_{des}, F_x and $F_{x,ref}$ denote the force and the force reference in the direction of x, respectively. The parameter M is the virtual mass and D the virtual damping of the impedance the robot is controlled to behave like (in the direction x). No position reference was used, as it was only interesting to control the force.

To make the controller safe to use, the maximum output velocity ($\dot{x}_{des}$) was limited. This limitation was made in such a way that no wind-up problems occurred. The switching between different control modes, e.g., from position to force control, was made by bumpless transfer, i.e., the new controller was initially being set to have the same control signal as the previous controller.

Choice of force control parameters

The controller parameters were chosen according to a pole placement design, of the poles in the transfer function from the force reference, F_{ref}, to the measured force, F. The controller (7.4) together with the contact model (7.1), where the location of contact is ignored, gives

$$\begin{cases} \ddot{x} = \frac{1}{M}\left(F - F_{ref} - D\dot{x}\right) \\ F = -K_{env}x - D_{env}\dot{x} \end{cases} \tag{7.5}$$

In (7.5) the velocity level control scheme of the assembly framework (Sec. 3.2) is used and the assumption of an ideal velocity controlled robot is made, i.e., $\dot{x} = \dot{x}_{des}$. The time-domain equations can be transformed to the frequency domain by the Laplace transform, giving

$$\begin{cases} s^2X(s) = \frac{1}{M}\left(F(s) - F_{ref}(s) - DsX(s)\right) \\ F(s) = -K_{env}X(s) - D_{env}sX(s) \end{cases} \tag{7.6}$$

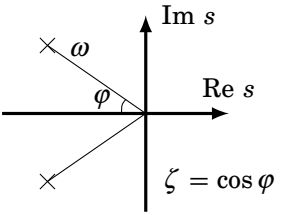


Figure 7.2 Illustration of the pole placement design (7.8).

By eliminating $X(s)$ in (7.6) the following relation between F_{ref} and F is achieved

$$F(s) = \frac{\frac{D}{M}s + \frac{K_{env}}{M}}{s^2 + \frac{D_{env} + D}{M}s + \frac{K_{env}}{M}} F_{ref}(s) \tag{7.7}$$

Hence, the measured force is related to the force reference by a second-order linear time-invariant dynamical system. A stable pole placement design for such a system can be parameterized according to Fig. 7.2, which gives the denominator polynomial

$$s^2 + 2\zeta\omega s + \omega^2 \tag{7.8}$$

Comparison of the coefficients of the denominator in (7.7) and the specification polynomial in (7.8) gives that the force control parameters should be chosen as

$$M = \frac{K_{env}}{\omega^2} \quad , \quad D = \frac{2\zeta K_{env}}{\omega} - D_{env} \tag{7.9}$$

where estimated values of the contact stiffness and the damping should be used.

The actual force controllers were implemented in discrete time, handled by discretization of the control law (7.4) (the sampling period used was 4 ms). The largest approximation is the assumption of neglected robot dynamics in the realization of the control law (7.4). This will only be approximately true up to a certain bandwidth, and the stability margins will depend on unmodeled dynamics, e.g., robot stiffness dynamics and time delays originating from sensor processing. The bandwidth of the force controller, ω, will thus have to be chosen with these considerations taken into account.

Torque control parameters

Torque control during assembly operations often means two or more point contacts. A change in the torque reference will therefore change the measured force, as there is a coupling between the measured force and torque. Usually the contact material for all contacts is approximately the same, which means that the same contact model that was identified during the first phase with only one contact can be reused. The remaining uncertainty is about the location of the second contact relative to the first, and this can be estimated, e.g., with an RLS estimator. Once the location of the contact is estimated, the formulas for controller parameters (7.9) can once again be used, with the stiffness $\hat{K}_{env}\hat{L}$ and the damping $\hat{D}_{env}\hat{L}$, where $\hat{L}$ is the estimated distance between the two contact points, and $\hat{K}_{env}$ and $\hat{D}_{env}$ are the estimated stiffness and damping, respectively.

Alternative specification of torque control

When performing assembly operations with two-point contacts, it is not always easy to choose appropriate set point values for the force and the torque controllers. Instead, an alternative is to control the force in each contact point. The estimation outlined earlier in this section gives the required information about the relative location of the contacts, i.e., the distance between them. This makes it possible to calculate the force originating from each contact, and transform force references in each contact to equivalent forces and torques.

This way of specifying the force and the torque during a two-point contact assembly operation will simplify the procedure for the user. The easiest way to implement it is to transform the two-force reference from the user, to force and torque references, and keeping separate control of force and torque. The user should, however, be presented with measurements transformed into forces from two contacts.

Assembly task

The assembly task considered as an example was the snapfit assembly scenario described in Sec. 3.2, with kinematic chains and the coordinate frames illustrated in Fig. 3.2. The assembly strategy used was the one displayed in Fig. 3.4.

The adaptation strategy described should only be used when the force control parameters are not well tuned, i.e., usually the first time the assembly is performed or when something has changed, e.g., at the use of a new gripper. The adaptation phases can be considered as separate states in between the nominal ones, see a part of the state machine implementing the sequence in Fig. 7.3, cf. Fig. 3.4.

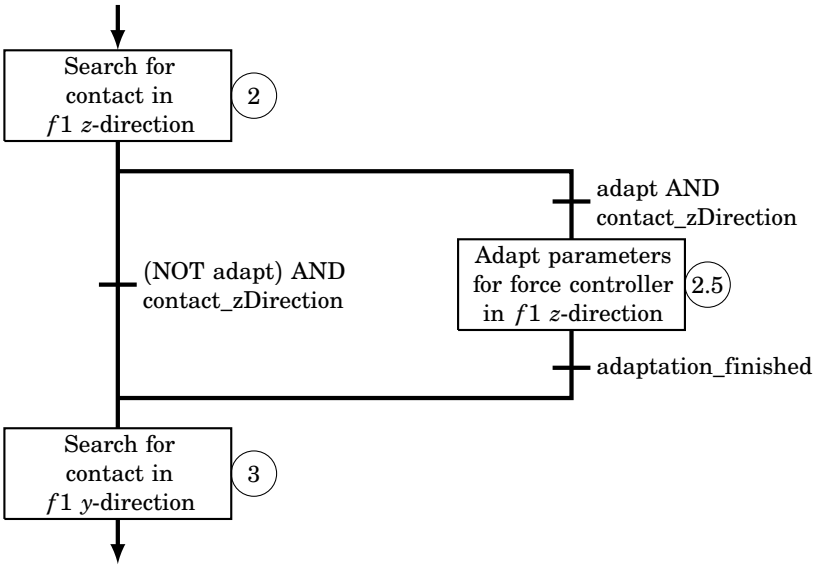


Figure 7.3 A part of the state machine implementing the assembly sequence. The parameter *adapt* decides whether the adaptation phase should be entered or not.

7.3 Experimental results

Contact with different materials

An experiment where contact was made with three different environments was used to test that the adaptation gave the desired performance. An initial search towards the environment was made until a contact force was detected. A force controller was then started with poorly tuned parameters, i.e., a default initial setting, and the adaptive algorithm was initiated. The force reference given was a square wave, and the forgetting factor λ was chosen to be 1, as the environment was not assumed to vary over time. Once the covariance of the contact model parameter estimates became low, the force control parameters were updated. A bandwidth of $\omega = 5\,\mathrm{rad/s}$ and a relative damping $\zeta = 0.8$ were chosen for the controller.

Experimental data are shown in Fig. 7.4, the left column shows the measured force and the force reference, and in the right column, the estimated stiffness and damping are displayed. In the top-most diagrams the environment was a soft plastic foam. The fact that the material was soft can be seen in the force response when contact was made, as the force is slowly built up. The nominal force control parameters were used in the first period of the reference signal, and the parameters were so

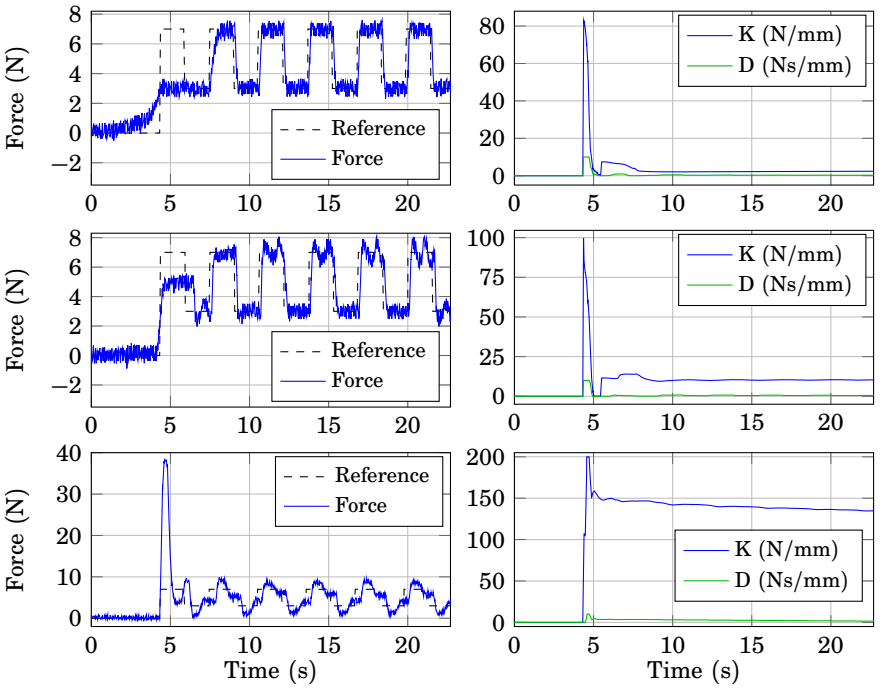


Figure 7.4 Experimental data from an experiment where contact was made with different environments. The top diagrams show contact with soft plastic foam, the middle diagrams contact with a mouse pad, and the bottom diagrams contact with a table surface. The left diagrams show the measured force in blue and the force reference in black. The right diagrams show estimated stiffness and damping.

poorly tuned that hardly anything happened. When the estimated contact parameters were used, however, the reference was satisfactorily tracked. The estimates of the contact stiffness and the damping can be seen to converge in less than 5 seconds.

The second environment used was a mouse pad, and results are displayed in the middle diagrams in Fig. 7.4. This material was stiffer, but both the control and estimation behavior was similar to the first case. The last environment was a table surface, and results are displayed in the bottom diagrams in Fig. 7.4. The initial force transient shows that this environment clearly was the stiffest. When the estimated contact parameters were used, the resulting control performance was worse than in the two previous cases. This was probably caused by that the assumptions made when deriving the control parameters were not completely

valid for the chosen control bandwidth and the stiffness of the contact material. Even though the performance is worse than for the previous environments, it is acceptable in regular assembly tasks.

The estimate of the stiffness started with a large transient for all environments, which was caused by the choice of a large initial covariance. Choosing it smaller, however, would lead to slower convergence for the parameter estimates.

Adaptation in an assembly sequence

The adaptation strategy was used to tune the force control parameters in an assembly sequence, experimental data are displayed in Figs. 7.5–7.8. Force data from the beginning of the sequence are shown in Fig. 7.5. State 2 was the search motion in the $f1$ z-direction, and the adaptation of the force control parameters for the z-coordinate was started in state 2.5[1], when contact was detected. The initial parameters were poor, as shown by the large initial force transient. On the other hand, the transient gave good excitation for the estimation algorithm. Initially, the force reference in state 2.5 was a sinusoid, to get a reference that would not be too hard for the poor controller to follow. Once the covariance of the estimate decreased below a threshold, the reference was switched to a square wave, to get more excitation. In order not to disturb the estimation algorithm, all other output directions were controlled to keep their current position during the adaptation phase. This phase was finished once the covariance decreased below a second threshold.

Search motions and adaptation in the $f2$ y- and x-directions then followed. Here it can be noted that the initial transients were much lower than for the z-direction and that the adaptation phases lasted somewhat longer. State 5 was the rotational search around the $f2$ x-axis, where the forces were controlled to be constant to keep the contact.

The identified contact model parameters and the norm of the P-matrix (a measure of the size of the covariance of the estimates) are shown in Fig. 7.6. It can be seen that the contact in the z-direction was considerably stiffer than in the other directions. The contact material itself had approximately the same properties in all directions, but the gripper and the switch was much stiffer in the z-direction than in the others. The slower convergence for the estimation in the y-direction can clearly be seen in the plot of the norm of P. Occasionally the algorithm gave unreasonable estimates, such as negative parameters, and to avoid problems with this the algorithm was supervised. The values used for chosing force controllers were projected into allowed intervals, see e.g., the damping

[1] The half states correspond to the adaptation states, e.g., as was illustrated for state number 2 in Fig. 7.3.

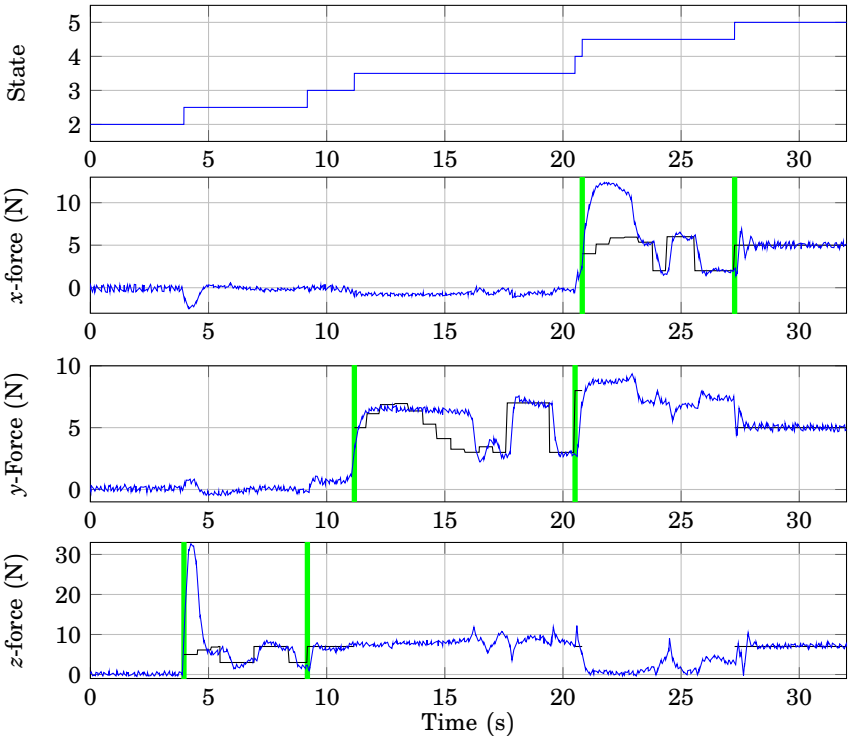


Figure 7.5 Experimental data from the beginning of the assembly sequence. The top diagram shows the state sequence, with state numbers defined in Fig. 3.4, and the decimal states defined according to Fig. 7.3. The remaining diagrams show the measured force (blue) and the force reference (black) for the coordinate directions in frame $f1$. The reference is only shown when the coordinate is force controlled. The adaptation phases are marked with vertical green lines.

parameter between $t = 11\,\text{s}$ and $t = 15\,\text{s}$. The estimation of the contact location, x_{env} in (7.1), is not shown because it is not relevant for the assembly sequence, but it also converged to a reasonable value for each contact model.

The search speeds in the assembly sequence had to be slow to handle the initial force control parameters, see e.g., the transient in the z-force in Fig. 7.5 at $t = 4\,\text{s}$. Once the control parameters had been tuned, it was possible to increase all search speeds.

The data shown in Figs. 7.5 and 7.6 have only been a one-point contact. The two-point contact was made in the second part of the assembly, see experimental data in Figs. 7.7 and 7.8. The adaptation for the torque

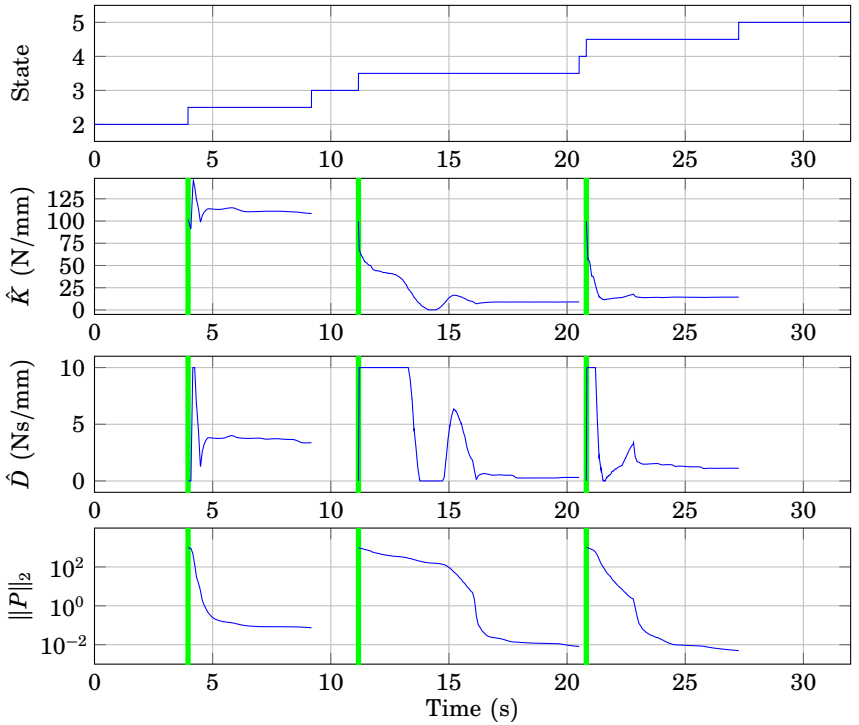


Figure 7.6 Experimental data from the beginning of the assembly sequence. The top diagram shows the state sequence, the second the stiffness parameter estimate, the third the damping parameter estimate, and the last diagram the norm of the P-matrix. The beginning of each adaptation phase is marked with a green line. The first phase is for the parameters in the z-direction, the second in the y-direction, and the third in the x-direction. Adaptation was only performed in the adaptation states, i.e., the half states.

controller (around $f2$ x-axis) started when the two-point contact was detected, i.e., when state 5.5 was entered. The resulting controller, active in the end of the adaptation phase, shows some overshoots when the reference is a square wave. This means that better reference tracking probably can be achieved by decreasing the control bandwidth, but this is not good for the performance in the assembly sequence, where it needs to react fast to disturbances caused by movements in other directions not to lose contact. The following action in the assembly sequence was to find the slot with the second contact point, by a search around the $f2$ z-axis. Once found, detected by a large z-torque, the switch was pushed down until

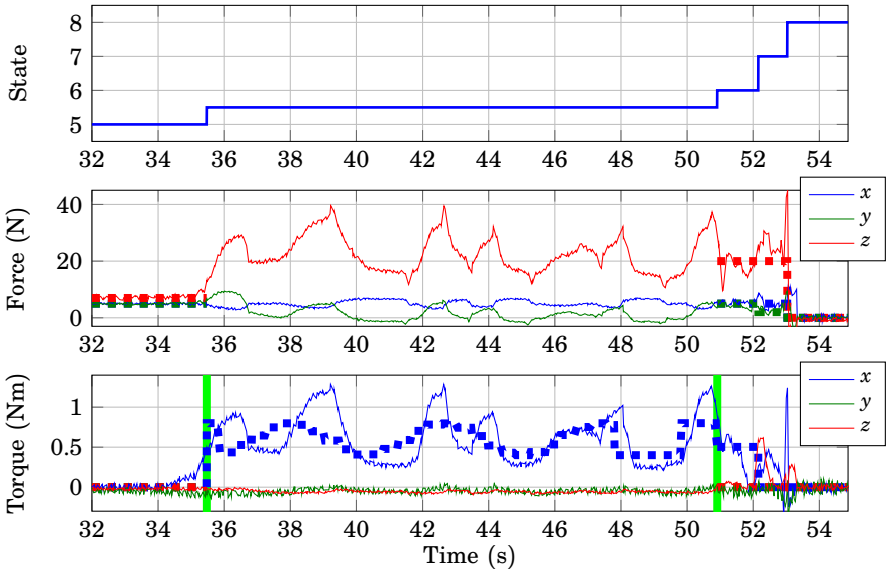


Figure 7.7 Experimental data from the final part of the assembly sequence. The top diagram shows the state sequence. The second diagram shows the measured force (solid lines) and the force reference (dashed lines) for the coordinate directions in frame $f1$. The third diagram shows the measured torque (solid lines) and the torque reference (dashed lines) around the coordinate axis of frame $f2$. The adaptation phase is marked with vertical green lines. Only the torque around the x-axis is controlled in the adaptation phase.

it was correctly inserted. Finally, the whole assembly was lifted to show that the sequence had finished.

The estimation of the distance between the two contact points is shown in Fig. 7.8. The estimate initially varies, and even becomes negative, which is handled by the previously mentioned supervision of the algorithm. A negative distance is further considered to be more of an issue than a negative damping parameter, so the P-matrix was also reset to a larger magnitude to restart the estimation. The estimate finally converges to approximately 34 mm, which is within 1 mm from the true value.

Alternative specification of torque reference

The approach where the user specifies two forces instead of one force and one torque in a two-point contact situation, described on page 109, was implemented in the assembly sequence. The only relevant state in the sequence was state 6, and experimental data from this state are shown in

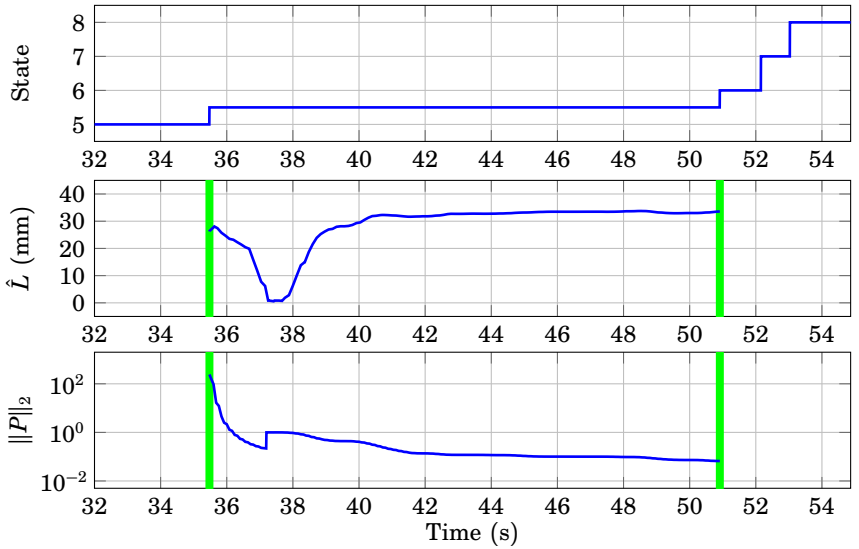


Figure 7.8 Experimental data from the final part of the assembly sequence. The top diagram shows the state sequence, the second diagram the distance parameter estimate (between the contact points), and the last diagram the norm of the P-matrix. The adaptation phase is marked with green lines.

Fig. 7.9. The first contact point was the end of the switch that first made contact, and the force reference for this point was set to 5 N, enough to not lose contact. It was desired that the other end of the switch slides down into the slot, and the reference was therefore a larger force, here 15 N. By using the identified distance between the two contact points, the given specification was translated to a force and a torque reference, see the two top diagrams in Fig. 7.9.

The control performance is good in the beginning of the time slot shown in Fig. 7.9. The references are lost in the end, and this was caused by the switch sliding down into the slot, i.e., contact was lost for the second contact point. The lost contact point can also be seen in the torque diagram, as the torque approaches zero. This behavior is an indication of a successful assembly.

7.4 Discussion

The adaptation algorithm described in Sec. 7.2 was successfully implemented on an industrial robot system. The achieved performance was

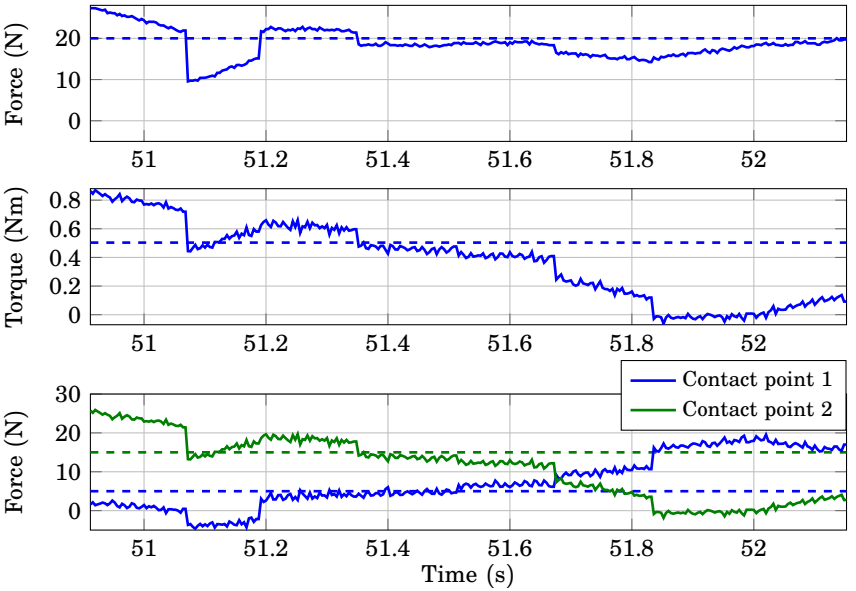


Figure 7.9 Experimental data from state 6 in the assembly sequence. The top diagram shows the z-force, the middle diagram the x-torque, and the bottom diagram the estimated equivalent forces acting on the two ends of the switch. Measured force/torque are shown with solid lines and references with dashed lines.

satisfactory, both for soft and stiff contacts, and it could be used to free the user from the tedious work of tuning the force controllers manually. Some performance degradation for stiff contacts was present that was not foreseen by the design procedure. This was caused by a too coarse approximation of the robot dynamics, by making the assumption of an ideal velocity controlled robot. To get a better control design, which considers the limitations of the robot system, also the robot dynamics have to be modeled.

An option that might enhance the control performance is to resort to an optimal controller, e.g., an LQG or H_∞-controller. But this means that the impedance control structure has to be abandoned, which might not be desirable. The impedance control parameters have a physical interpretation that might be valuable, e.g., in an error situation.

In this work, only decoupled contact models and force controllers were considered. This was convenient concerning the use of force controllers in the assembly framework, but using coupled contact models and force controllers might be a way to increase the control performance. One dif-

ficulty with this approach is the identification phase, it will be hard for the system to autonomously know when it is possible to identify a coupled model, i.e., when the robot is in contact in several directions. The solution might be to include this information in the task specification.

The adaptation was implemented as separate states in the controlling state machine. A dedicated excitation signal was used to assure that input data to the estimation algorithm were sufficiently exciting. The use of an excitation signal might disturb the assembly process and the involved components, but as a low force amplitude was used and as the robot was not moving during the adaptation phase, it was assumed that the continuation of the assembly process would be unaffected by the adaptation phase. A further development of the adaptation algorithm could be to run the adaptation in each assembly operation without a dedicated excitation signal.

The contact locations have been estimated during the assembly sequence, but this information was not used. One way to use it is to decrease the search times, by increasing the velocity of search motions when being far from the identified contact locations, i.e., using the strategy that was described in Chap. 4.

The method of using two forces instead of one force and one torque in a two-point contact scenario simplifies the task specification for the user, as the coupling between the force and the torque can be ignored. Generalizing the strategy to more than two-point contacts is hard, as the conversion from force and torque measurements to multiple forces is very hard or even impossible to solve.

The stop criterion used for finishing the adaptation phases in this work was based on thresholding the norm of the P-matrix (7.3). These thresholds will have to be chosen with respect to the noise level from the force sensor, which might be difficult to do before running the adaptation algorithm. Another option would be to instead rely on the rate of change of the norm of the P-matrix, i.e., stopping the adaptation phase when this rate becomes lower than a threshold.

The stability of the system presented has not been addressed. An attempt to prove stability would need approximations of the dynamics of the robot and the servo control system, and then the impact of these approximations would have to be considered as well. In this work, a supervision approach was taken to make sure that the system did not become unstable. By making sure that all estimated parameters stayed within predefined bounds and that the velocity of the robot was limited. Further, if the measured contact force exceeds a threshold, the robot will stop and require assistance from an operator.

To the best of the author's knowledge, a similar approach has not been previously presented within assembly. Adaptive force control with compa-

rable results has been performed, e.g., in [Roy and Whitcomb, 2002] and [Kröger et al., 2004]. They both show similar results for corresponding contact stiffnesses, but this thesis also considers significantly stiffer contact environments. A stiffness of over 100 N/mm was estimated in Fig. 7.4, compared to a stiffness around 20 N/mm in [Kröger et al., 2004] and below 1 N/mm in [Roy and Whitcomb, 2002].

7.5 Conclusions

A method for self-tuning of force controllers to use in industrial robots has been described. It was based on identification of a contact model using an RLS algorithm. The force controller considered was an impedance controller and its parameters were chosen according to a pole placement design. The method was implemented on an industrial robot system and used in an assembly task.

8

Robotic Force Estimation without Force Sensor

8.1 Introduction

The traditional way of programming industrial robots is to use position control and follow predefined trajectories, using feedback from the joint position sensors. Modern robot controllers are very good at this and perform these tasks both fast and with high accuracy. In tasks where the robot has to physically interact with the environment, however, this control strategy is less advantageous. The accuracy of the robot and the location and geometry of everything in the workspace have to be known with high precision, and this is usually hard to achieve. A remedy to this problem is to introduce additional sensing, e.g., a force sensor that gives the robot capabilities to handle position uncertainties by sensing the contact forces. A force sensor can thus be used to make the robot system more robust towards uncertainties.

Force sensing can be achieved in a number of different setups. One alternative is to use a wrist-mounted sensor, another option is to use a sensor that is mounted in the workspace, i.e., not on the robot. A third approach is to have torque sensors in each joint of the robot, e.g., as was done in the DLR light-weight arm [Albu-Schäffer et al., 2007]. The main drawbacks with using force sensors are that they usually are very expensive and may add unnecessary mass to the system, and in the case of a sensor in the workspace, force sensing will only be available when the robot is in contact with the sensor.

An alternative to using a force sensor is to estimate the external forces applied to the robot based on sensing already available in the robot. Usually this includes position sensors in the joints and torques exerted by the motors, or the motor currents. The main problems with estimating forces is how to handle the large disturbances that are present, e.g., originating

from friction. Further problems arise if there are gearboxes in the robot, since a high gear ratio will scale down external forces applied to the arm side when measured on the motor side. This is tightly connected with the notion of backdrivability; a robot is backdriveable if it is possible to move it when the motors are off. Backdrivability has been identified as an important property for human-robot interaction [Krebs et al., 2004]. A too high gear ratio will make it really difficult to overcome the friction in the motors, and it will make the robot in practice non backdriveable and it will also be very difficult to estimate any external forces.

One example of a force estimation method using the motor currents is [Simpson et al., 2002], where also friction disturbances were carefully modeled. Another approach using the motor torques is [Wahrburg et al., 2014]. It is also possible to use the motor torques together with a dynamical model of the robot to estimate the forces. In [Van Damme et al., 2011] it is presented how this can be performed by using a filtered dynamic model and a recursive least-squares estimator.

Another approach is to use some kind of observer. One way is to use disturbance observers, i.e., to use a dynamical model of the robot and consider deviations from this as disturbances caused by external forces, see, e.g., [Eom et al., 1998] and [Ohishi et al., 1992] for examples of this approach. Direct force observers can also be used. In [Ohishi et al., 1991] an H^{∞} force observer was used, and in [Murakami et al., 1993; Ohishi, 1993] torque observers together with dynamic models were used. In [Hacksel and Salcudean, 1994] and [Alcocer et al., 2003] the force was estimated by considering how position estimation errors behaved as a damped spring-mass system.

Force estimation can also be performed by using adaptive methods. In [Jung et al., 2006] a method based on the extended Kalman filter together with an adaptive law was presented. Another adaptive force estimation approach is given in [Sararoody et al., 2005]. Estimation of the robot joint velocities and accelerations together with a dynamic model are used to perform impedance control without a force sensor in [Tachi et al., 1990].

A clear disadvantage with the proposed methods using observers and adaptive methods is that they usually require a dynamic model of the robot. Such a model is straightforward to derive in theory, but in practice you often do not know the values of all parameters involved. It is possible to perform identification experiments, but identifying all of the dynamic parameters for a manipulator with six or seven joints is difficult, as the experiments to be performed must be chosen with care to be sufficiently exciting to make all parameters identifiable. The friction in the joints is also hard to model in a good way. These difficulties seem to be hard to overcome, as almost no force estimation method based on observers or adaptive methods presents experiments using a robot with more than

three joints.

Another type of model-based approach is based on using the generalized momentum, e.g., presented in [De Luca and Mattone, 2005] and [De Luca et al., 2006]. The benefit of using the generalized momentum is that joint acceleration measurements are not needed, but the method still requires knowledge of a dynamical model for the robot. The generalized momentum method has mostly been used for collision detection, and experimental results with the DLR lightweight robot was presented in [De Luca et al., 2006]. An extension of the method is presented in [Wahrburg et al., 2015], where the external forces are modeled as a linear system such that the forces can be estimated with a Kalman filter.

Methods for force estimation and sensorless force control are also available in commercial robot systems, e.g., both Toshiba [Toshiba, 2015] and ABB [ABB Robotics, 2011] provide such products. These systems are, however, designed to work well for large forces and they are therefore not suitable in small-parts assembly, as is considered in this thesis.

In previous work, e.g., [Murakami et al., 1993; Ohishi, 1993; Simpson et al., 2002], it was assumed that the joints were always moving, and no attention was given to the large uncertainties in the friction torques at velocities close to zero. The performance of force estimation can be improved by modeling the friction carefully and considering position-dependent torque variations, e.g., see [Popovic and Goldenberger, 1998; Simpson et al., 2002]. In [Du and Nair, 1999; H. Olsson et al., 1998] modeling of low-velocity friction phenomena are considered. These models, however, require knowledge of many parameters that are challenging to identify and prone to change, due to, e.g., temperature and wear.

This chapter will propose two different methods of how to perform force sensing without a force sensor, by instead estimating the forces from the joint position control errors or the joint motor torques. The methods are experimentally verified in a number of experiments, including real-world small part assembly tasks. A comparison of the methods is also presented. The main focus of this chapter is on force estimation for the light-weight YuMi robot, but also force estimation for a standard industrial robot, the IRB140, is considered and experimentally evaluated.

8.2 Robotic force estimation using joint position-control errors

Method

This section will present an approach that is possible when each joint on the lowest level is individually controlled, which is a common solution in industrial robots. By disabling the integral action in the joint controllers,

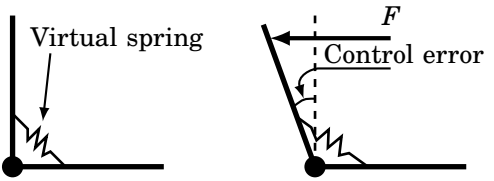


Figure 8.1 An illustration of what happens when an external force F is applied to a joint of a robot. The applied force will give rise to a position control error.

they will act as virtual springs, and the deviation of each joint angle from its reference will correspond to a joint torque, see illustration in Fig. 8.1. Due to friction and gravity, the joint errors may become large if the integral action is removed completely, leading to bad performance in the position control loops and bias in the force estimate. One remedy to this problem is to use a small integral part, which allows force transients to be detected, but over time the position errors will be removed. Estimation of forces based on joint errors, using small intergal action, acts as a high-pass filtered version of the forces.

The joint torques τ and the end effector forces F are related by

$$\tau = J^T F + e \tag{8.1}$$

where $J = J(q)$ is the robot Jacobian, q is the robot joint coordinates, and e are disturbance joint torques with the assumption $\mathbb{E}[e] = 0$ and $\mathbb{E}[ee^T] = R_e$. The minimum variance estimate of the force is then given by $\hat{F} = (JR_e^{-1}J^T)^{-1}JR_e^{-1}\tau$ according to the Gauss-Markov theorem [Kailath et al., 2000], but if the disturbances are large, the estimate may be of very poor quality. By adopting a Bayesian approach and using prior knowledge about the particular assembly operation, it may be possible to improve the force estimates. Assume that the prior knowledge about F can be described by $\mathbb{E}[F] = \bar{F}$ and $\mathbb{E}[(F - \bar{F})\,(F - \bar{F})^T] = R_F$, and that the distribution of τ conditioned on F is given by (8.1), then the minimum variance unbiased estimate of F is [Kailath et al., 2000]

$$\hat{F} = (JR_e^{-1}J^T + R_F^{-1})^{-1}(JR_e^{-1}\tau + R_F^{-1}\bar{F}) \tag{8.2}$$

For example, it may be known that the contact torques on the end effector may be very small during an assembly operation. By reflecting this knowledge in R_F, the estimates of the contact forces can be improved.

Calibration

The robot used in experiments was the ABB YuMi. The force estimation performance is affected by how the detuning of the joint controllers has

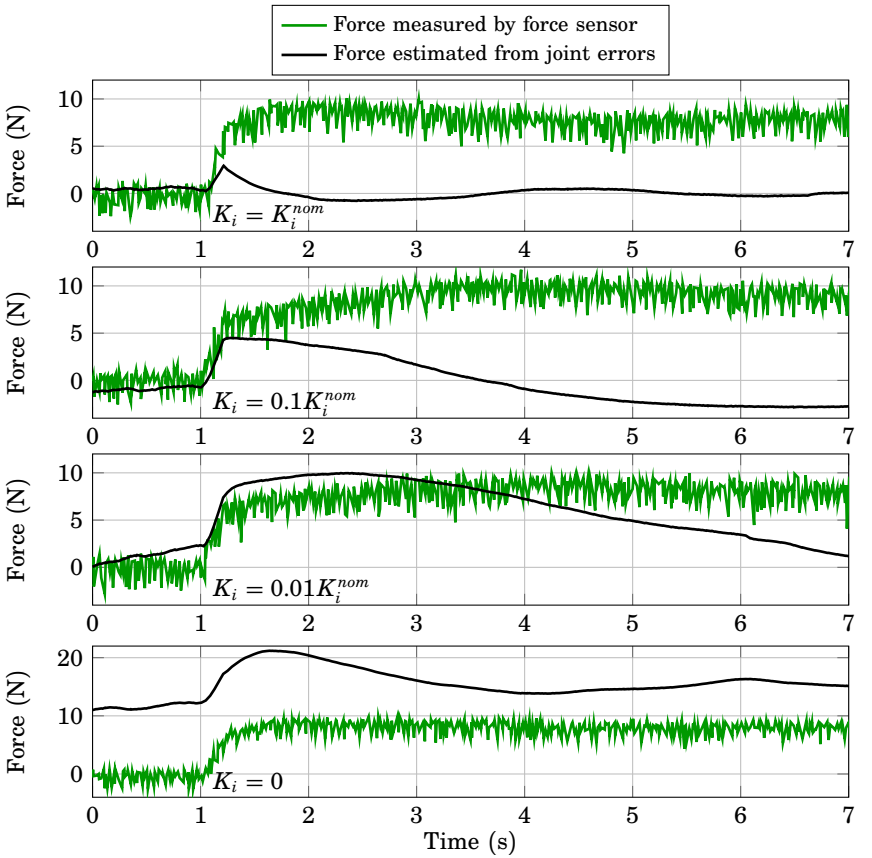


Figure 8.2 Estimated force (black) and measured force (green) in one direction for some different values of the integral gain in the joint controllers. The nominal controllers have the integral gain K_i^{nom}. The topmost diagram has integral gain K_i^{nom}, the second diagram $0.1K_i^{nom}$, the third diagram $0.01K_i^{nom}$, and the lowermost no integral action at all.

been performed. If the integral part is completely removed there will be problems with offsets, because of gravity and friction forces. Keeping the integral part, however, makes it impossible to estimate a constant force, as this would require the joint controllers to have a stationary error. Keeping the integral action will act as a high-pass filter on the estimated forces, which means that only transients can be detected. The behavior for different detunings is shown in Fig. 8.2, where the force sensor was used to find contact in one direction and control the contact force to a constant value. It can be seen in the diagrams that a high integral gain gives a

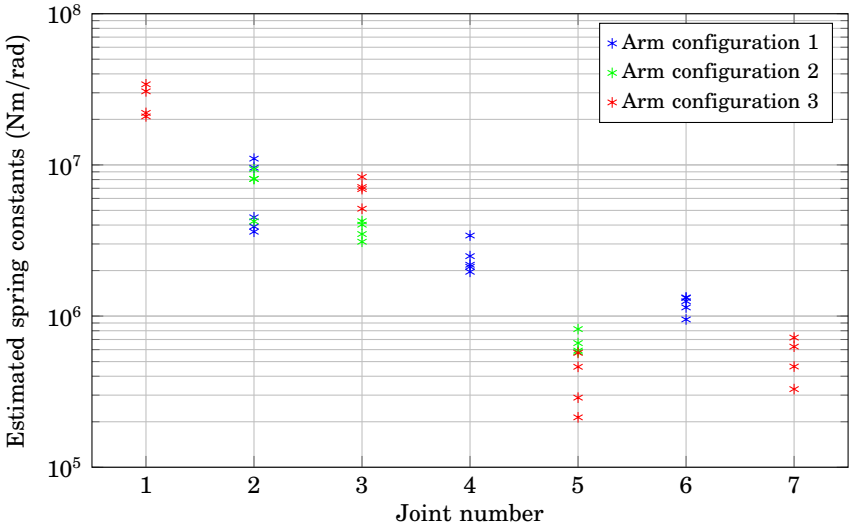


Figure 8.3 Experimentally determined spring constants for the different joints. The joints are numbered from shoulder to wrist. The different colors denote values obtained from different arm configurations.

transient with a short duration, which may be hard to detect. Removing the integral action completely, however, introduces a bias in the estimate. The final controller detuning chosen to be used in the assembly task was with $K_i = 0.03K_i^{nom}$ as integral gain, where K_i^{nom} is the integral gain in the nominal joint controllers.

A large disturbance when doing force estimation is friction in the joints. Experiments were performed to estimate the friction magnitude in each joint, which mostly consisted of Coulomb friction. These values were used to choose the diagonal elements of R_e, the variance of the disturbance forces in Eq. (8.1). The effect of gravity was assumed to vary slowly, such that the remaining integral part in the joint controllers could compensate for it.

According to the identified joint friction torques, they will lead to force estimation errors with an order of magnitude of 1 N. Estimation errors of this size were measured for the experimental execution of the assembly task, see Fig. 8.5.

To determine the spring constants of the joints, forces or torques were applied to the tool of the robot, and the amplitude of the resulting joint error transients were recorded. Doing this for three different arm configurations, it was possible to determine the stiffness of all joints. Approximately five experiments were performed for each arm configuration, and

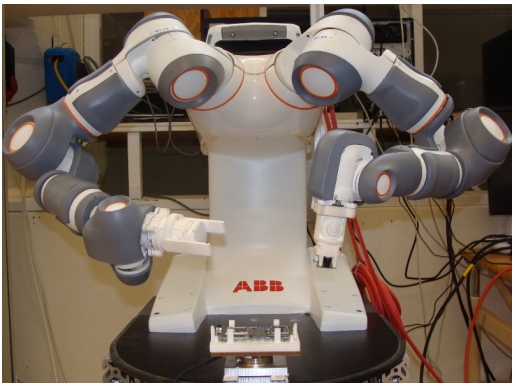


Figure 8.4 The setup for the shield can assembly task. The force/torque sensor, mounted beneath the fixture, was only used for verification and evaluation of the estimated forces.

the results can be seen in Fig 8.3. For each joint, the mean value of the experiments was later used for force estimation.

Assembly task

The force estimation method was used to accomplish the shield can assembly task, described in Sec. 3.4, using the left arm of YuMi, see experimental setup in Fig. 8.4. A six degrees-of-freedom ATI Mini40 force/torque sensor, mounted beneath the fixture for the printed circuit board (PCB), was used to get validation data.

Experimental results

The assembly strategy that was described in Sec. 3.4 had to be modified when the estimated force was used. The high-pass character of the force estimates made it impossible to control constant forces. As an alternative, once a search motion made contact, the position was controlled instead of the force. This modification of strategy made the assembly less robust, but the effect was small concerning the uncertainties in this particular task, with the PCB being placed in a well-defined fixture and the main uncertainties being the position of the shield can in the gripper. As the contact torque estimates were found out to be unreliable, the rotational search in state 7 was replaced with a position controlled motion to the estimated final position of the shield can. To be able to do this maneuver successfully it had to be assumed that the mounting plane of the PCB was known with good accuracy, which was a reasonable assumption to make, as the PCB was placed in a fixture. Gripping uncertainties, corresponding to small rotations around the $f2$ z-axis, were not expected to be a problem,

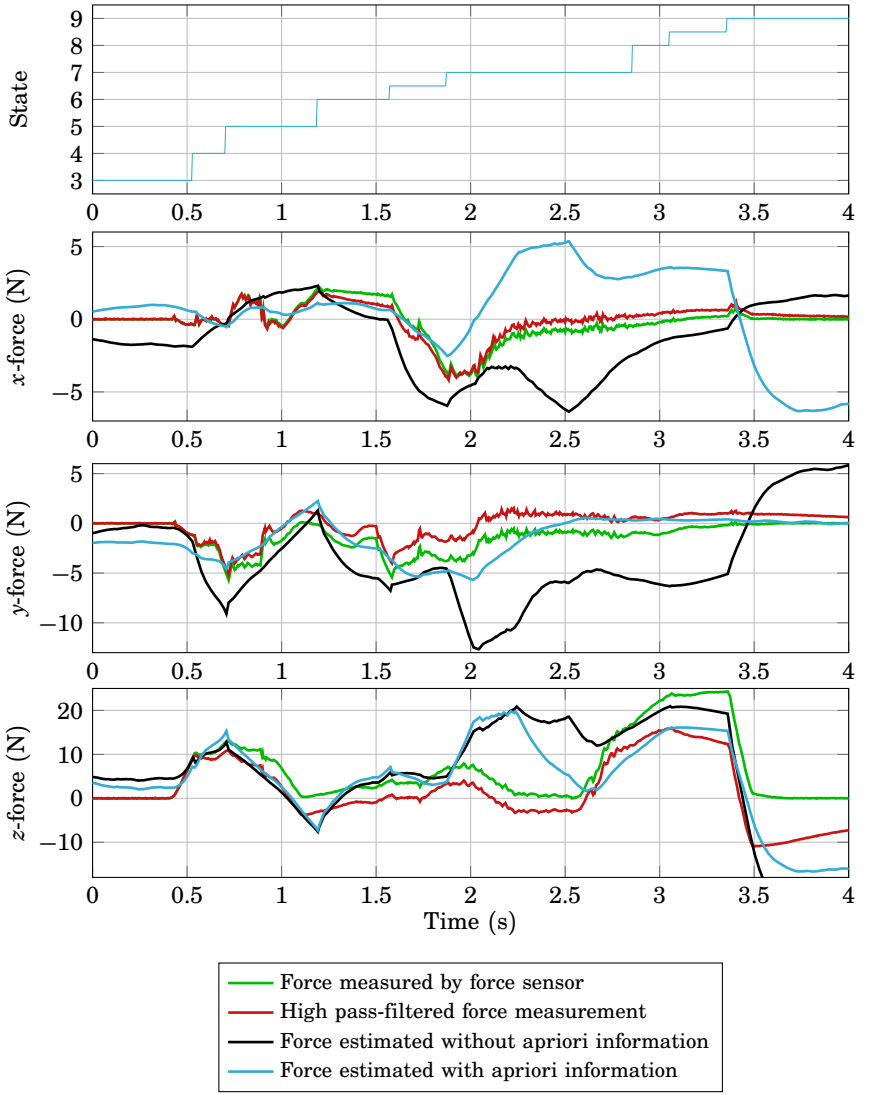


Figure 8.5 Measured and estimated forces from the assembly sequence without force sensor together with the state sequence. Forces estimated with a priori information about the low torques are shown in blue, and forces estimated without a priori information in black. For reference, raw force data from the force sensor is shown in green, and high-pass filtered force data in red. The data from the force sensor have not been used for control in this execution.

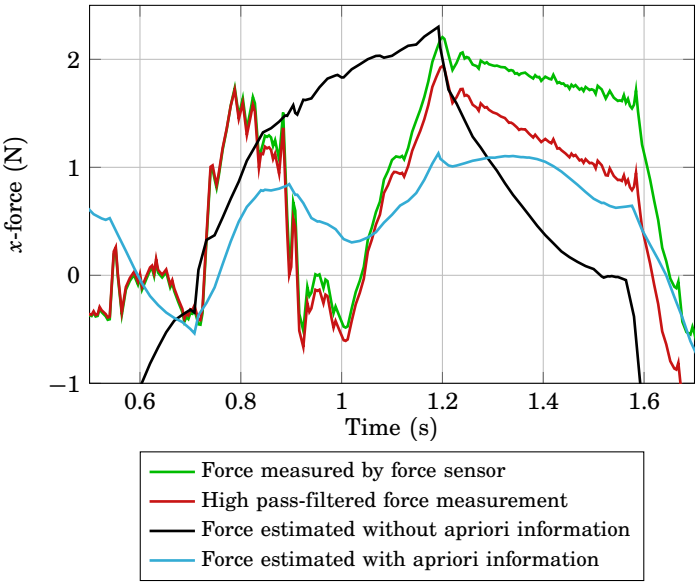


Figure 8.6 Selected part of the measured and estimated x-force around the transition from state 5 to 6.

as the gripper was compliant in this direction and because the shield can was rotated down when in contact with a corner of the socket, such that the shield can was forced onto the socket by its edges. Once the rotating motion was finished, the robot pressed the shield can with a large force towards the socket to be certain that the shield can was assembled correctly. The robot kept pressing for 0.3 s and then the assembly was assumed to be finished.

Force data from an experimental execution are given in Fig. 8.5. The high-pass character of the estimated force is verified by including a high-pass filtered version of the measured force. Two versions of the estimated force are shown, with and without a priori information about the low contact torques. The first state shown is the search for contact in the $f1$ z-direction. The transition condition, a large positive z-force can be seen in all force curves at $t = 0.5$ s. The following state is the search in the positive y-direction and it makes contact at $t = 0.7$ s, which is seen by a large negative y-force. State 5 is then a search in the x-direction. The search motion was made with contact in both the z- and the y-directions, and this initially caused a friction peak in the x-force (at $t = 0.8$ s), the relevant part of the x-force is displayed in Fig. 8.6. The force then disappears and the contact was made at $t = 1.1$ s. The estimated force with a priori

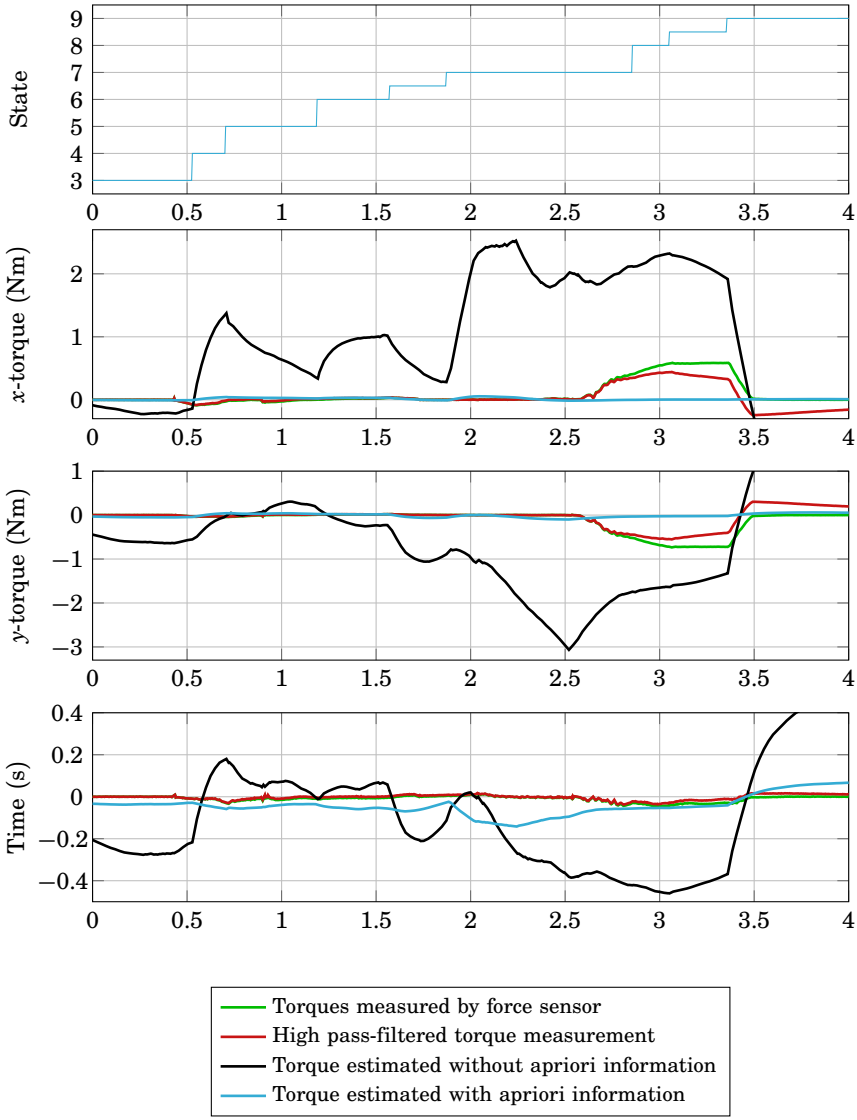


Figure 8.7 Measured and estimated torques from the assembly sequence without force sensor together with the state sequence. Torques estimated with a priori information about the low torques are shown in blue, and torques estimated without a priori information in black. For reference, raw torque data from the force sensor is shown in green, and high-pass filtered torque data is shown in red.

information shows the same behavior as the measured force, but the force estimate without a priori information does not. The transition to the next state was finally made at $t = 1.2\,\text{s}$. The final search for the corner of the socket was then made in two steps; first a y-search in state 6 and then an x-search in a new state, here called 6.5. The transition condition for the y-search can be seen at $t = 1.6\,\text{s}$ and the transition condition for the x-search at $t = 1.9\,\text{s}$. The transitions can be seen in both estimated forces, but the resemblance with the measured force is better for the estimate with a priori information. State 7 is the position control of the orientation, such that the shield can was rotated down onto the socket. The rotation was made around the origin of frame $f2$. Modeling errors in the position of this frame was the reason for the large z-forces around $t = 2.7\,\text{s}$, as the rotation was not made exactly around the origin of $f2$. These forces were detected and the reference position in the z-direction was adjusted. The shield can was pressed onto the socket with a large force in state 8, which can be seen in the z-force at $t = 3.0\,\text{s}$. Finally, the robot waited 0.3 seconds in state 8.5 and then moved away in state 9.

Measured and estimated contact torques from the experimental execution are given in Fig. 8.7. It can be seen from the sensor measurements that torques significantly different from zero only are present during the last stage of the assembly, i.e., during state 7 and 8. The estimate with no a priori information is really bad, neither the magnitude nor the shape show any resemblence with measured data. Using the a priori information gives a reasonable magnitude on the estimate, but it does not react to the applied torques in state 7 and 8 and the estimate is therefore unreliable.

The empirical distribution of the force estimation errors during the execution of the assembly scenario for the case when a priori information about the size of the contact torques were used is displayed as a histogram in Fig. 8.8. It can first be noted that the errors can not be explained as Gaussian noise. The magnitude of the errors is quite large concerning the force levels during the assembly. The estimated forces were reasonably correct when there was contact, but the performance was worse when no contact was present, see, e.g, after $t = 3.4\,\text{s}$ in Fig. 8.5, which partially explains the large portion of errors far from zero in Fig. 8.8. Another explanation for the estimation errors was the presence of friction in the joints. The similarity between the high-pass filtered force data and the force estimate, as can be seen in Fig. 8.5, at least in the case when a priori information was used, verifies the high-pass character of the estimate.

Discussion

Estimating forces from the joint control errors instead of using a force sensor introduces some difficulties in the implementation of the assembly

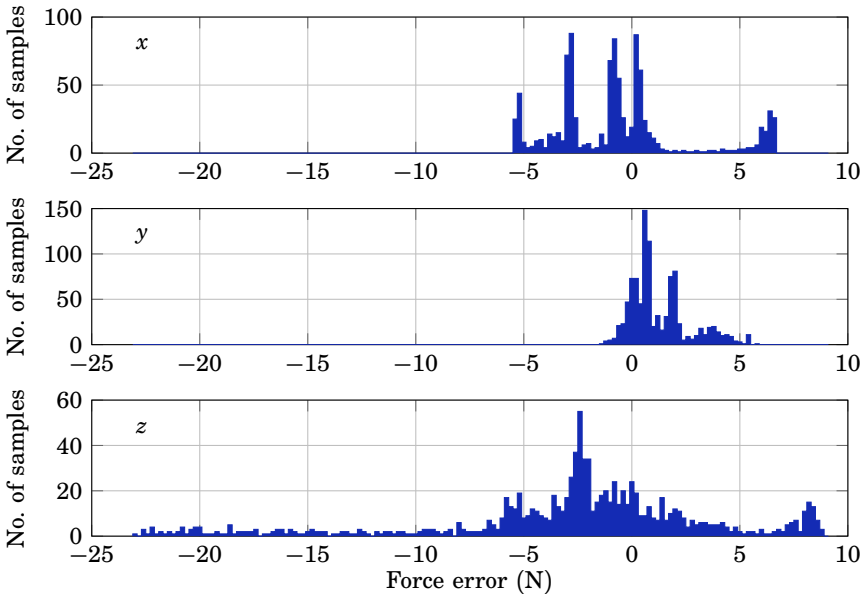


Figure 8.8 Histogram of the force estimation errors in Fig. 8.5 (for the force estimated using a priori information about the low torques). Each bin in the histogram has a width of 0.2 N.

operation, as compared to using a force sensor. Doing it the way presented in this thesis requires you to choose an appropriate detuning of the joint controllers. Since the disturbances in the estimates may be quite large, special care must be taken when choosing force thresholds in the design of the assembly sequence. The force estimation method was not very good at estimating the correct force level, it was, however, better at detecting the transients from when different contact situations were established, which made it possible to accomplish the assembly task.

When the robot is not moving, the Coulomb friction in the joints makes it particularly hard to estimate the forces, since the contribution from gravity and other disturbance forces is unknown, and it is very difficult to predict how much additional torque is needed in the different directions to overcome the friction and make the joint move. When the robot is moving, however, the Coulomb friction torque is constant and even a small external force (e.g., caused by a collision) can affect the motion and be seen as a transient in the joint control-errors. Since the disturbances from the friction are very similar between different executions of the same motion, the situation becomes even better and it is possible to robustly detect forces with the same order of magnitude as the friction disturbances.

Since the disturbances are velocity dependent, however, there may be a need to retune the force thresholds if the speed of motion is changed.

When moving at high speeds, dynamic effects and lag in the position tracking may cause large errors in the force estimation, but sometimes increasing the speed of motion makes the sensing easier, since transients caused by collisions then become easier to detect in the high-pass filtered data.

The fact that the disturbances to a large extent are systematic, indicates that adaptation or learning techniques could be successful in improving the performance. By further on considering the entire signal instead of its instantaneous value it is probably possible to find more robust transition conditions. Another set of parameters that possibly can be adapted is those concerning the detuning of the joint controllers.

In the shield can assembly scenario, the sensing problem is very hard, since the contact forces are in the same order of magnitude as the disturbances caused by friction in the joints. To get useful estimates of the forces, the contact torques had to be assumed to be very small. In a different scenario, where contact forces are much larger than the friction disturbances, it should be possible to perform assembly without assuming that the contact torques are small.

8.3 Robotic force estimation using motor torques and modeling of low-velocity friction disturbances

Introduction

In this section, a method for force estimation using the motor torques is presented. The method is based on modeling the friction disturbances in the joints and taking this into account when calculating the force estimate.

The focus of the method presented in this section is not on estimating the friction torques of the individual joints rigorously, but on modeling the velocity-dependent uncertainties in the friction torques and combining measurements from multiple joints to compute an accurate estimate of the contact force. In particular, the noise in the velocity measurement is taken into account, and it is modeled that the Coulomb friction is quite well known when a joint is moving, but has much larger uncertainty for velocities close to zero. The force is estimated by solving a convex optimization problem, and an approximate confidence interval is also calculated.

The validity of the approach is investigated by comparing the estimated forces with measurements from a force sensor. The method is finally tested in a dual-arm screwing assembly task, described in Sec. 3.3, with the setup shown in Fig. 8.9.

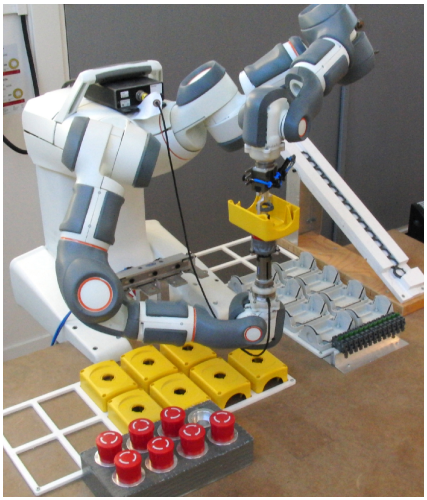


Figure 8.9 The setup for the screwing assembly task. The force/torque sensor, mounted at the wrist of the robot's right arm, was only used for verification and evaluation of the estimated forces.

Motivation Measured motor torques are often available measurements in a robot system, but they contain large disturbances. An example of measured motor torque from a dual-arm assembly task execution is displayed in Fig. 8.10. The right robot arm was controlled to be still while the left robot arm was manipulating an object held by the right robot arm. The actual external torque, as measured by a force sensor, is also displayed in the figure. It can be seen that the disturbances are larger than the signal of interest. It can further be seen that the disturbances are different for the two arms, with a distinct Coulomb friction pattern for the arm that is controlled to move, while the arm that is controlled to be still appears to have less predictable disturbances.

Method

Modeling The method used for force estimation in this section is based on the measured joint motor torques. The model used is

$$\tau_m = \tau_{grav} + \tau_{dynamic} + \tau_{ext} + \tau_e \tag{8.3}$$

where τ_m denotes the measured motor torques, τ_{grav} denotes the torques originating from gravity, $\tau_{dynamic}$ denotes dynamic torques originating from accelerations of the robot, τ_{ext} denotes external torques, and τ_e denotes disturbances due to, e.g., friction, measurement noise, and modeling errors.

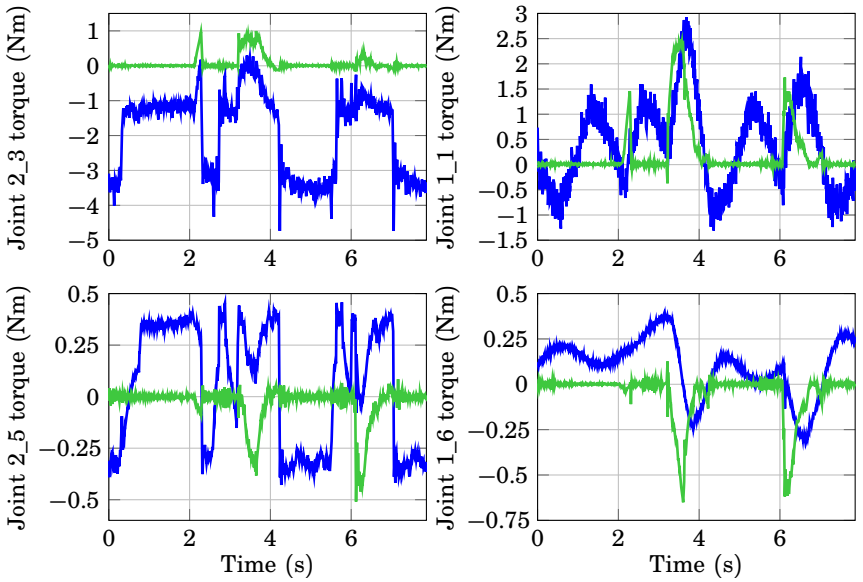


Figure 8.10 Measured motor torque and applied external torque from an assembly sequence. The diagrams in the left column show data from the left arm, which was controlled to move, and the diagrams in the right column show data from the right arm, which was controlled to be still. The upper diagrams show data from base joints, and the lower diagrams show data from wrist joints[1]. The joints chosen for display are those where the external torques were the most visible. It can be seen that the disturbances were as large or even larger than the signal of interest.

The influence from gravity, τ_{grav}, can be calculated if the mass and center of mass are known for each link of the robot. If they are not known, it is fairly easy to perform identification experiments to find these parameters. The actual calculation is, e.g., described in [Siciliano et al., 2009]. The dynamic torques, $\tau_{dynamic}$, can also be calculated if the dynamic parameters of the robot are known, i.e., moment of inertia for each link of the robot. The dynamic torques will, however, be small in tasks where it is interesting to use force estimation, as the robot will be interacting with the environment and thus needs to move quite slowly. It is therefore assumed that the dynamic torques can be neglected.

[1] With joint numbering according to the ABB convention, joint 2_3 is joint 2_7, joint 2_5 is joint 2_4, and joint 1_6 is joint 1_5

The external joint torques originate from external forces and torques applied to the robot. If it is assumed that all external forces are applied to the end effector of the robot, the external joint torques are given by

$$\tau_{ext} = J^T F \tag{8.4}$$

where $J = J(q)$ is the Jacobian of the robot, q the joint coordinates, and F denotes the force/torque applied to the end effector.

Disturbance torques The disturbance, τ_e, influencing each joint mostly consists of Coulomb friction, which can be modeled to give the following contribution for joint i

$$\tau^i_{Coulomb} = \begin{cases} \tau^i_{C,max}, & \dot{q}_i > 0 \\ \tau^i_{C,min}, & \dot{q}_i < 0 \end{cases} \tag{8.5}$$

where $\dot{q}_i$ denotes the velocity of joint i, and $\tau^i_{C,max}$ and $\tau^i_{C,min}$ denote the constant friction levels for positive and negative velocities, respectively. What happens at zero velocity is not given by the model, and the torque might be anywhere in the interval $[\tau^i_{C,min}, \tau^i_{C,max}]$. Therefore, for low velocities close to zero, the Coulomb friction contribution can be modeled by a uniform random variable.

Another type of friction is viscous friction. It can be modeled to give the following contribution for joint i

$$\tau^i_{viscous} = c_i \dot{q}_i \tag{8.6}$$

where c_i is a constant specific for each joint.

Another large disturbance is measurement noise, which can be modeled to have a zero-mean Gaussian distribution.

Disturbance model To find out the disturbance characteristics, an identification experiment was performed for each joint. The joint was then moving back and forth with a low piecewise constant acceleration, without any external forces applied to the robot. Two versions of this experiment are displayed in Fig. 8.11; the upper diagram shows an experiment with low velocities, and the lower diagram an experiment with higher velocities. The raw data, sampled at 250 Hz, were low-pass filtered with the discrete-time filter (8.7) to remove some of the noise influence.

$$H(z) = \frac{0.4}{1 - 0.6z^{-1}} \tag{8.7}$$

The Coulomb friction is easy to see in both experiments, but other effects such as stiction or Stribeck friction [H. Olsson et al., 1998] is not visible. As was suggested earlier, the amount of friction for zero velocities

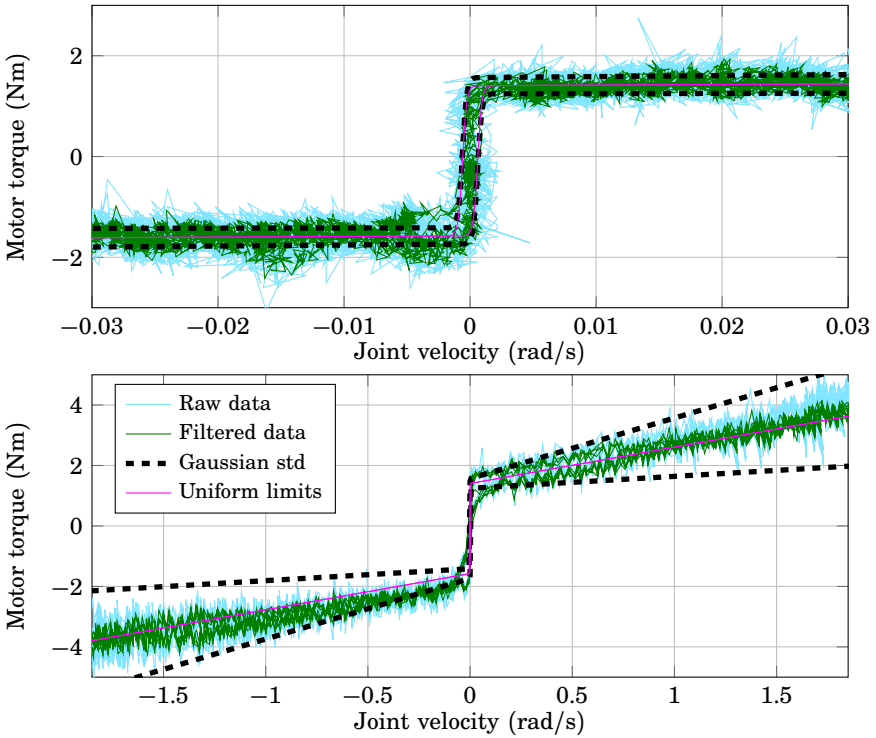


Figure 8.11 Motor torque data from an experiment where one joint of the robot was controlled to move back and forth with piecewise constant acceleration. The upper diagram shows an experiment with only low velocities, and the lower diagram shows an experiment with higher velocities. The disturbance characteristics are clearly visible in this experiment. Also shown is one standard deviation of the measurement noise multiplied with a velocity dependent factor, and the upper and lower limits for the uniform distribution describing the Coulomb and the viscous friction.

varies between $\tau_{C,min}$ and $\tau_{C,max}$, and due to noise in the velocity measurements this is true also for measured velocities slightly different from zero. Aside from Coulomb friction, the experiment with large velocities exhibits viscous friction. Further, there is also noise present.

A probabilistic model of the disturbances is therefore that the Coulomb and the viscous friction are the outcome of a uniform random variable with a velocity-dependent range. This range is zero for large velocities and the range grows for low velocities. One way to describe this range is by using sigmoid functions for describing the upper and the lower limits. To incorporate also viscous friction, a linear term is added. The upper

and lower limit for each joint can be described by the following functions (joint index skipped)

$$\begin{aligned}\tau_{f,max}(\dot{q}) &= \tau_{C,min} + \frac{\tau_{C,max} - \tau_{C,min}}{1 + e^{-A(\dot{q}+B)}} + c\dot{q} \\ \tau_{f,min}(\dot{q}) &= \tau_{C,min} + \frac{\tau_{C,max} - \tau_{C,min}}{1 + e^{-A(\dot{q}-B)}} + c\dot{q}\end{aligned} \tag{8.8}$$

The parameter A determines the slope of the sigmoid function, and the parameter B the width of the area between the curves. Parameters for such functions were manually tuned for each joint of the robot, and an example is seen as magenta curves in Fig. 8.11.

A Gaussian noise term is used to account for measurement noise, uncertainty in the friction limits and unmodeled disturbances. It can be seen in Fig. 8.11 that the variance of the noise increases when the velocity increases. The model used is therefore that the variance of the noise is velocity dependent and the standard deviation for different velocities is calculated as the standard deviation for low velocities multiplied with a factor $(1 + k|\dot{q}_i|)$, where $k \geq 0$ is a parameter. One standard deviation of the noise is displayed in Fig. 8.11. Data recorded during assembly operations indicated that the actual disturbances at high velocities were higher than the measured data in Fig. 8.11 indicate. Hence, the one-standard-deviation limit may appear overly pessimistic in this figure.

To conclude, the total disturbance torque is modeled as

$$\tau_e = \tau_f + e \tag{8.9}$$

where $\tau_{f,min}(\dot{q}) \leq \tau_f \leq \tau_{f,max}(\dot{q})$, and e is zero-mean Gaussian with diagonal covariance matrix $\mathbb{E}[ee^T] = R_e(\dot{q}) = \text{diag}(\mathbf{1} + k|\dot{q}|)^2 R_e^{slowvel}$, where $R_e^{slowvel}$ is the noise covariance for slow velocities, and $\mathbf{1}$ is a vector of ones with the same dimension as $\dot{q}$.

Force estimation Let $\bar{\tau}$ be the motor torques compensated for gravity, calculated as

$$\bar{\tau} = \tau_m - \tau_{grav} \tag{8.10}$$

Using (8.3), (8.4), (8.9), and the assumption that the dynamic torques are negligible, this gives

$$\begin{aligned}\bar{\tau} &= \tau_{ext} + \tau_e \\ &= J^T F + \tau_f + e\end{aligned} \tag{8.11}$$

where $\bar{\tau}$ and J are given, and τ_f and e are random variables with uniform and Gaussian distributions, respectively. The ML (Maximum Likelihood)

estimate of F is then given by the F solving the following optimization problem (see derivation in App. B)

$$\begin{aligned} \underset{\text{over } F,\tau_f}{\text{minimize}} \quad & \tfrac{1}{2}\left(\bar{\tau} - J^T F - \tau_f\right)^T R_e^{-1}\left(\bar{\tau} - J^T F - \tau_f\right) \\ \text{subject to} \quad & \tau_{f,min} \leq \tau_f \leq \tau_{f,max} \end{aligned} \tag{8.12}$$

The estimate given by (8.12) can be improved by adopting a Bayesian approach and using prior knowledge of F in the particular task. The type of prior knowledge that can be used is, for instance, that the contact torques are small compared to the torque disturbances, and by reflecting this knowledge in the distribution of F it is possible to improve the quality of the estimated contact forces. This is the same strategy that was used in Sec. 8.2.

Assuming that the prior on F is Gaussian with $\mathbb{E}[F] = \bar{F}$ and $\mathbb{E}[(F - \bar{F})(F - \bar{F})^T] = R_F$, and that F and e are uncorrelated, the maximum a posteriori (MAP) estimate of F is given by the F solving (see derivation in App. B)

$$\begin{aligned} \underset{\text{over } F,\tau_f}{\text{minimize}} \quad & \tfrac{1}{2}\left(\bar{\tau} - J^T F - \tau_f\right)^T R_e^{-1}\left(\bar{\tau} - J^T F - \tau_f\right) \\ & + \tfrac{1}{2}\left(F - \bar{F}\right)^T R_F^{-1}\left(F - \bar{F}\right) \\ \text{subject to} \quad & \tau_{f,min} \leq \tau_f \leq \tau_{f,max} \end{aligned} \tag{8.13}$$

The optimization problem (8.13) is convex and can be solved numerically in real time, as described on page 141.

Confidence interval estimation The uncertainty of the estimate given by (8.13) varies significantly with, e.g., the velocity of the different joints and the robot Jacobian. Hence, it is important to calculate the uncertainty of every estimate individually.

It is difficult to compute exact quantiles for the solution of (8.13), but this section describes a method for extracting approximate confidence intervals that can be computed in real time. The method is first described for the case with a single robot joint without prior, and then generalized to handle multiple joints and a prior distribution on F.

One-dimensional case The proposed confidence interval for the case of a single robot joint with no prior and the Jacobian $J = 1$ is illustrated in Fig. 8.12. The limits are calculated as

$$\begin{aligned} \tau_{conf,min} &= \tau_{f,min} - \lambda\sigma \\ \tau_{conf,max} &= \tau_{f,max} + \lambda\sigma \end{aligned} \tag{8.14}$$

where σ is the standard deviation of the Gaussian tails and λ is a parameter deciding the confidence level of the confidence interval.

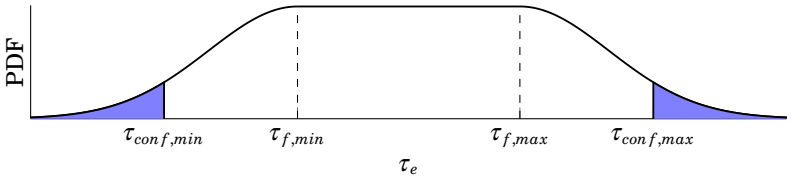


Figure 8.12 Illustration of the proposed confidence interval on the probability density function (PDF) of $\tau_e = \tau_f + e$, a flat part and two Gaussian tails. The blue areas indicate the portion of the measurements expected to be outside the confidence interval.

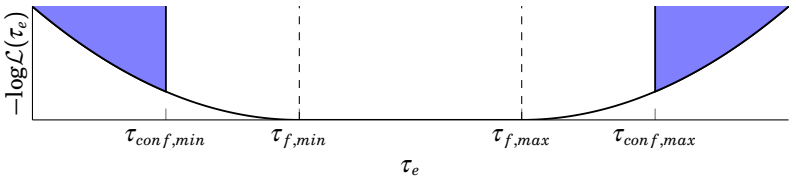


Figure 8.13 The log-likelihood for τ_e, where the probability density function was illustrated in Fig. 8.12.

For the special case where $\tau_{f,min} = \tau_{f,max}$ (the distribution of τ_e is Gaussian), the portion of the measurements outside the confidence interval is $2(1 - \Phi(\lambda))$, where $\Phi(\cdot)$ is the cumulative distribution function of the zero-mean unit-variance Gaussian distribution.

An alternative way of finding the limits (8.14), is to minimize the negative log-likelihood function of τ_e and adding a gradient to push the solution toward the upper or lower limit. The procedure is illustrated in Fig. 8.13, which shows the negative log-likelihood of τ_e, and in Fig. 8.14, where the addition of the gradient is illustrated. The log-likelihood of a zero-mean Gaussian with standard deviation σ is given by

$$\log\mathcal{L}(e) = -\frac{e^2}{2\sigma^2} + \text{const.} \tag{8.15}$$

$$\frac{d}{de}\left(\log\mathcal{L}(e)\right) = -\frac{e}{\sigma^2} \tag{8.16}$$

Hence, at the limits of the confidence interval, the derivative of the negative log-likelihood function of τ_e is

$$-\frac{d}{de}\left(\log\mathcal{L}(\pm\lambda\sigma)\right) = \pm\frac{\lambda}{\sigma} \tag{8.17}$$

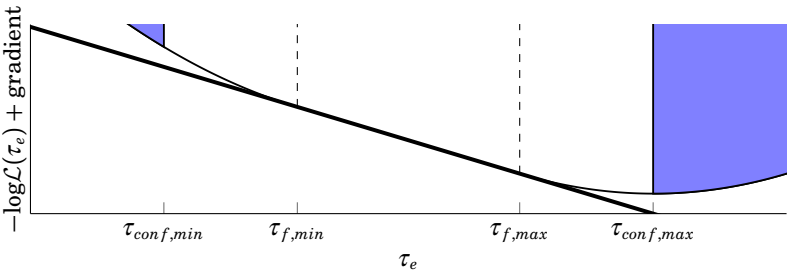


Figure 8.14 Illustration of the log-likelihood of τ_e added with a gradient (the thick black line). The upper limit of the confidence interval is the minimum of this function. By instead adding the gradient with reversed sign, the lower limit of the confidence interval will be the minimum of the function.

Consequently, adding a gradient with one of the slopes (8.17) to the negative log-likelihood of τ_e and finding the minimum, gives one of the limits (8.14) as the solution. This way of calculating the limits is described because it generalizes to higher dimensions better than (8.14).

Multi-dimensional case For the multi-joint problem (8.13), first assume that $\tau_{f,min} = \tau_{f,max}$, (i.e., Gaussian distribution). The standard deviation σ of the marginal distribution of F in the direction of the unit vector v is then given by

$$\sigma = \sqrt{v^T \left(JSR_e^{-1}J^T + R_F^{-1}\right)^{-1} v} \tag{8.18}$$

where S is the identity matrix for the Gaussian case but may have other values for the general case, as described later in this section. The limits of the confidence interval in the direction v are then given by the F solving

$$\begin{aligned} \underset{\text{over } F,\tau_f}{\text{minimize}} \quad & \tfrac{1}{2}\left(\bar{\tau} - J^T F - \tau_f\right)^T R_e^{-1}\left(\bar{\tau} - J^T F - \tau_f\right) \\ & + \tfrac{1}{2}\left(F - \bar{F}\right)^T R_F^{-1}\left(F - \bar{F}\right) \mp \tfrac{\lambda}{\sigma} v^T F \\ \text{subject to} \quad & \tau_{f,min} = \tau_f = \tau_{f,max} \end{aligned} \tag{8.19}$$

where the "$-$" in the "$\mp$" is for the upper limit, and the "$+$" is for the lower limit. This formulation is obtained by adding the gradient (8.17) to the problem (8.13).

Returning to the general case, when $\tau_{f,min} \neq \tau_{f,max}$, some of the joints may get an estimated τ_e in the range $\tau_{f,min} < \tau_e < \tau_{f,max}$. The cost function for that joint is then locally flat, cf. Fig. 8.12, and should not be considered

when calculating (8.18). The optimization problem to solve becomes

$$\begin{aligned} \underset{\text{over } F,\tau_f}{\text{minimize}} \quad & \tfrac{1}{2}\left(\bar{\tau} - J^T F - \tau_f\right)^T R_e^{-1}\left(\bar{\tau} - J^T F - \tau_f\right) \\ & + \tfrac{1}{2}\left(F - \bar{F}\right)^T R_{\bar{F}}^{-1}\left(F - \bar{F}\right) \mp \tfrac{\lambda}{\sigma} v^T F \\ \text{subject to} \quad & \tau_{f,min} \leq \tau_f \leq \tau_{f,max} \end{aligned} \tag{8.20}$$

To find out for which joints the estimated τ_e ends up in the Gaussian part of the distribution, the following algorithm is proposed. It is assumed that n joints are used for force estimation and that R_e is diagonal.

1. Set $S = 0_{n\times n}$
2. Calculate (8.18)
3. Solve (8.20)
4. For the joints where $\tau_f = \tau_{f,min}$ or $\tau_f = \tau_{f,max}$, set the corresponding diagonal elements of S to 1
5. If S was modified in step 4, go to step 2. Else quit.

The intuition behind the above algorithm is the following. The problem (8.20) is first solved using a gradient based only on the prior. If the Coulomb friction for all joints is large, the resulting τ_e may all be within the flat part of the distribution and only the prior is used for determining the confidence interval. If any of the estimated τ_e reaches the Gaussian parts of the distributions, the gradient based only on the prior will not be able to push the estimate far down the Gaussian tails. The value of σ in (8.18) is then modified to include all joints where the τ_e estimate is in the Gaussian part, resulting in a steeper gradient, which may in turn push the estimate of additional joints to the Gaussian part of the distribution. The process (steps 2–5) is iterated until convergence.

Implementation

The optimization problem (8.13) is a convex optimization problem of fairly small size and can be solved in real time in a reliable manner. To this purpose, CVXGEN [Mattingley and Boyd, 2012] was used. It is a code generator for embedded convex optimization. The generated code is library-free C code, and this code has been connected to the robot controller via an Ethernet connection.

The generated solver is run on a Linux PC and the computation time to arrive to a solution is in the order of 0.3 ms. The robot controller is run with a sampling time of 4 ms, and the speed of the solver is therefore

sufficient to be run in each sample. It may, however, be difficult to calculate confidence intervals for all force/torque components, but this problem can be avoided by only calculating a subset of them. The solution with the Ethernet connection introduces a delay of one sample, as the indata to the solver is sent one sample before the solution is returned.

Assembly task

The force estimation method was used to accomplish the screwing in the emergency stop button assembly task, described in Sec. 3.3, using the YuMi robot. A six degrees-of-freedom ATI Mini40 force/torque sensor, mounted on the wrist of the right arm, was used to get validation data. The experimental setup is displayed in Fig. 8.9.

Experimental results

Calibration An experiment where the robot was programmed to slowly move around in its workspace was performed to identify the parameters used for calculating the gravity torque, τ_{grav}. The resulting parameters resulted in a mean absolute error ranging from 0.05 Nm for the wrist joints to 0.3 Nm for the base joints.

The friction model parameters were tuned by performing experiments of the type that were displayed in Fig. 8.11.

Force estimation The force estimation method was tested in an experiment where forces were applied to a static arm (controlled not to move). The estimated forces and confidence intervals are displayed in Fig. 8.15, together with validation data from the force sensor. All confidence intervals in this chapter were estimated with $\lambda = 1.96$ in (8.20), which would give a 95 % confidence interval for a Gaussian random variable. The parameter k was set to 5 s/rad.

Fig. 8.15 shows that the estimated force tracked the measured force well, but the confidence intervals seem to be overly pessimistic. The Coulomb friction is, however, a very large disturbance for low velocities. When the robot is moving, the uncertainty in the Coulomb friction is much smaller, as can be seen on the magenta-colored curve in the upper diagram of Fig. 8.11. Large external forces make the robot move slightly, and this gives significantly tighter confidence intervals than when the robot is still, see, e.g., the z-force at $t = 15$ s and $t = 18$ s in Fig. 8.15.

The prior used in this experiment was defined by

$$\bar{F} = 0_{6\times 1} \quad R_F = \text{diag}(10\,\text{N}, 10\,\text{N}, 10\,\text{N}, 0.1\,\text{Nm}, 0.1\,\text{Nm}, 0.1\,\text{Nm})^2 \tag{8.21}$$

such that no mean force and only small contact torques were expected. The benefit of using the prior is shown in Table 8.1, where the mean absolute

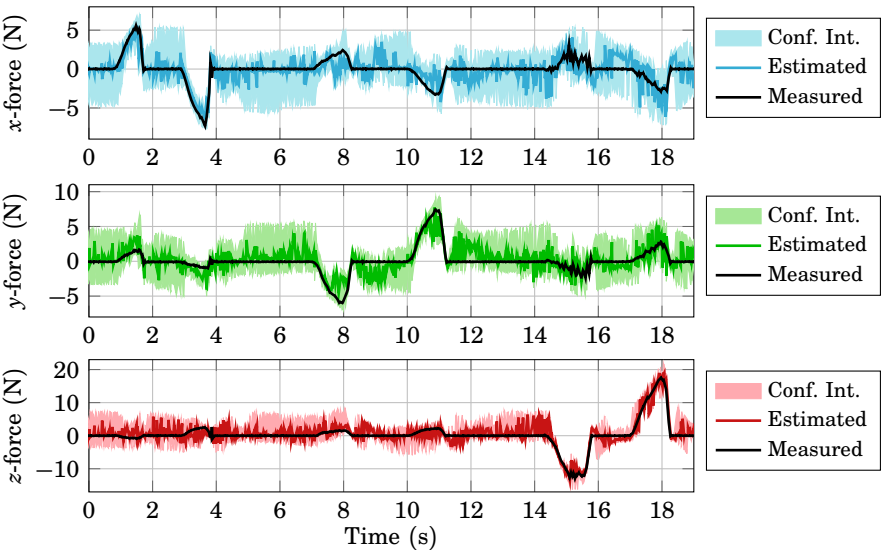


Figure 8.15 Data from an experiment where forces were applied to the end effector of the robot. A wrist-mounted force sensor was used to get validation data, which are displayed as black solid curves. The estimated force is displayed together with a confidence interval.

Table 8.1 Mean absolute estimation errors (N) for the experiment in Fig. 8.15.

Force direction	x	y	z
With prior	0.67	0.69	0.87
Without prior	1.58	1.22	2.12

estimation errors with and without the use of the prior are listed, showing significantly decreased estimation errors when the prior was used.

Screwing assembly task Estimated and measured forces from an execution of the screwing assembly task are displayed in Fig. 8.16. The forces are given in the coordinate frame $f1$ illustrated in Fig. 3.13. It can be seen that the estimated forces tracked the measured forces, at least when the measured forces were non-zero, i.e., during contact operations. When the measured force was zero, however, the estimated force was sometimes a bit wrong, e.g., in the y- and the z-directions around $t = 1.6\,\text{s}$. This estimation error was most likely due to modeling errors, as the robot was

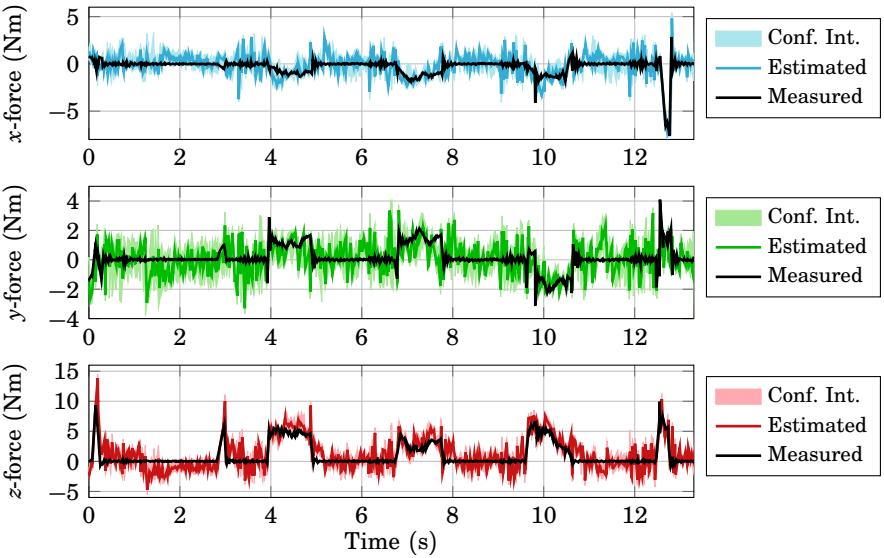


Figure 8.16 Measured and estimated forces from an execution of the screwing assembly task. Forces are given in the coordinate frame $f1$ illustrated in Fig. 3.13. The computed confidence intervals are also shown.

moving quite fast in this part of the assembly, but it was known that the movement would be performed in free space when the robot was moving fast, and therefore it was not that important to get a perfect force estimate.

For the assembly task, some of the important forces to detect were the contact forces in the z-direction when the nut was put on the thread. They were correctly detected at $t = 0.2\,\text{s}$ and $t = 3\,\text{s}$, and the confidence interval was tight at these moments. The screwing was finished when a large side force was detected at $t = 12.6\,\text{s}$; a zoom-in view on this part of the data is displayed in Fig. 8.17. It can be seen that the force estimate was both quite correct and confident when the forces occurred. Some oscillations can be seen in the force estimate, e.g., in the z-force around $t = 12.2\,\text{s}$. This might be caused by unmodeled disturbances, like cogging torques in the motors or mechanical resonances.

The estimated contact torques together with those measured with the force sensor are displayed in Fig. 8.18. For most samples, the confidence intervals for the estimated contact torques included the zero torque, since the friction torques were very large in comparison to the contact torques. Consequently, larger contact torques would be required for the estimator

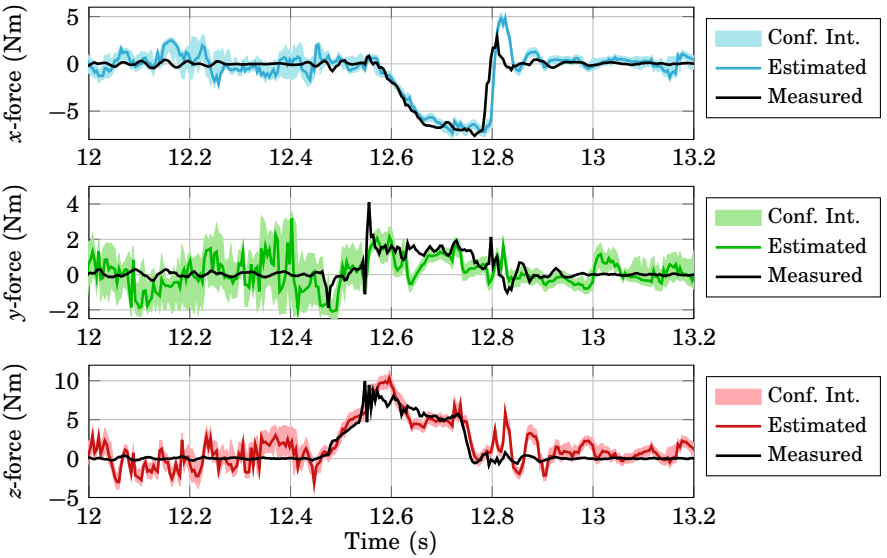


Figure 8.17 A zoom in on the measured and estimated force data from Fig. 8.16.

to be able to detect them reliably.

The force estimates presented in Figs. 8.16–8.18 were calculated using data from both arms, and in Table 8.2 they are compared to estimates based on only the right (static) or left (moving) arm. It can be seen that the estimate using the static arm, gave the most measurements within the confidence interval. These data should, however, only be used to evaluate the quality of the confidence intervals, not the force estimates. When the joints were not moving, the large uncertainty in the Coulomb friction resulted in wide confidence intervals, and hence many measurements were inside the confidence intervals. When the robot was moving, however, the model seemed to be a bit too confident about the estimate. Only looking at the samples when external forces were present did not change much.

The lower part of Table 8.2 shows the mean absolute estimation error. Here it can be seen, that when all samples were considered, the estimates from the static arm were the best, but only slightly better than the estimates using both arms. It may be surprising that using only one arm can give better results than using both arms, but when only the static arm was used, the confidence interval was large and usually enclosed the prior, which was $\bar{F} = 0_{6\times1}$ in this example. Hence most estimates were pulled to the prior, which was almost equal to the actual forces for most

Table 8.2 Statistics for the force estimation in the screwing assembly task. The upper half shows the percentage of the samples where the measured force was within the computed confidence interval. The lower half shows the mean absolute estimation error (forces in (N), torques in (Nm)). The left part shows statistics for when the entire assembly sequence is considered, while the right part only considers those samples when the measured force was non-zero, i.e., when the arms were in contact.

	All samples			Samples when external force was non-zero		
	Percentage of ground truth force/torque measurements within confidence interval					
Wrench component	Static arm	Moving arm	Both arms	Static arm	Moving arm	Both arms
x force	96.2 %	66.7 %	78.6 %	91.2 %	65.2 %	70.7 %
y-force	95.6 %	64.4 %	82.4 %	90.0 %	64.2 %	74.8 %
z-force	91.7 %	61.9 %	58.4 %	81.2 %	67.5 %	57.2 %
x-torque	97.1 %	72.2 %	87.4 %	93.2 %	64.7 %	80.5 %
y-torque	97.9 %	80.2 %	80.5 %	95.3 %	85.8 %	78.4 %
z-torque	99.8 %	77.4 %	72.8 %	99.6 %	89.9 %	77.5 %
	Mean absolute error					
Wrench component	Static arm	Moving arm	Both arms	Static arm	Moving arm	Both arms
x-force	0.60	1.08	0.60	1.05	1.23	0.62
y-force	0.63	1.40	0.64	1.14	1.42	0.64
z-force	0.91	1.38	1.09	1.33	1.29	1.02
x-torque	0.066	0.22	0.095	0.16	0.25	0.096
y-torque	0.036	0.15	0.11	0.073	0.13	0.099
z-torque	0.014	0.050	0.074	0.036	0.057	0.091

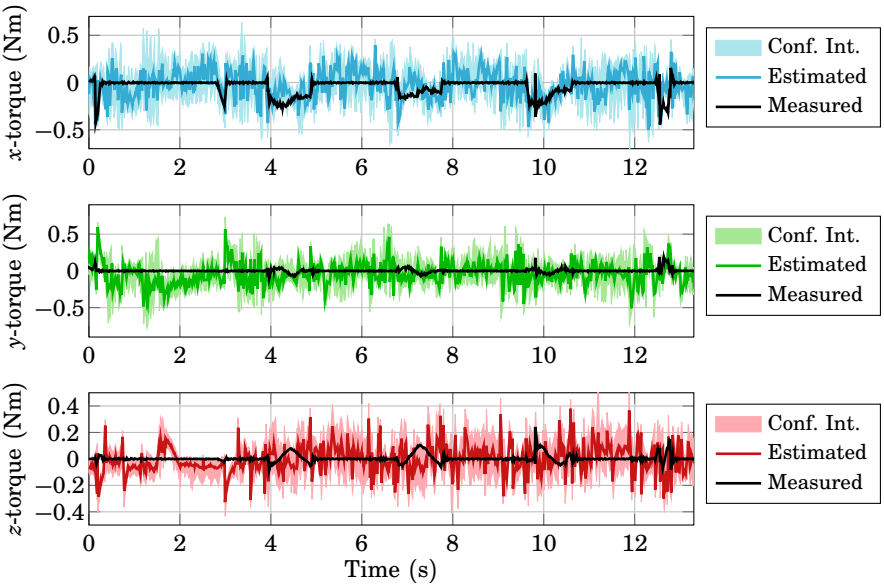


Figure 8.18 Measured and estimated contact torques from an execution of the screwing assembly task. The computed confidence intervals are also shown.

samples in this sequence. When there were external forces present, which is the situation when the force estimation is really useful, the estimates based on both arms were significantly better than those based on any single arm, as seen in the lower right part of Table 8.2.

The prior knowledge used in the assembly scenario was that the contact torques should be quite small, but not zero. This prior knowledge gave a slight increase in estimation performance compared to not using a prior, but it was not dramatic. More is gained in scenarios with only a point contact, where it is known that the contact torques should be zero, such as the experiment where forces where applied to the static robot arm.

Discussion

Experiments showed that, when the arms were in contact, complementing motor torque data from a moving arm with data from a static arm, significantly improved the quality of the estimated forces. The static arm could not have been exploited with previously published force estimation methods, which do not account for the uncertainty of the Coulomb friction

at low velocities.

The magnitudes of the contact torques in the experiments in this section were small compared to the uncertainties in the torque estimates, both with and without the use of a prior distribution on F. This was mainly caused by the relatively large disturbances. The contact torques considered were in the order of 0.1–0.3 Nm, which was in the same order of magnitude as the errors in the gravity compensation. Also errors in the Coulomb friction modeling gave disturbances in the same order of magnitude as the contact torques. Estimating forces was more beneficial, as relatively small forces could give rise to relatively large joint torques through long lever arms.

Some of the parameters used for force estimation in this section were manually tuned, including A and B in (8.8), the low-pass filter (8.7), and the velocity dependence of the Gaussian noise term. This should be possible to do in an automatic fashion, i.e., make experiments of the type displayed in Fig. 8.11 and choose the parameters by optimizing some criterion. This would further simplify the use of the method and it is considered as some of the future work.

This method for force estimation can also be used when the robot is in singular configurations, i.e., when the Jacobian, J, is singular. The force estimate will then rely on the prior, but the confidence interval will be huge in the singular direction, i.e., reflecting the uncertainty of the estimate.

A similar method to the one presented in this section is described in [Wahrburg et al., 2014]. The same approach is used, but with some differences. The most important difference is that the authors do not take the low velocity friction uncertainties into account, they only rely on a simple friction model with Coulomb and viscous friction. In this way, the force estimate can be calculated through matrix calculations, basically (8.2), and the optimization problem (8.13) does not have to be solved. Another main difference is that the covariances R_e and R_F are chosen to match data by solving an optimization problem using data from an experiment where the robot moves around in its workspace with a known load. In this thesis, these covariances were chosen with respect to the estimated noise levels and knowledge about which magnitudes of forces and torques that were expected.

8.4 Comparison of methods

The force estimation methods presented in this chapter, that is the joint control error method in Sec. 8.2 and the motor torque method in Sec. 8.3, were compared in two different experiments. Aside from the performance

of the estimators, also the detuning of the servo controllers was evaluated. The first experiment was to manually apply forces to a static robot arm, and the second experiment was an assembly task, namely putting the gray box in the fixture in the emergency stop button use case, described in Sec. 3.3. The left arm of the YuMi robot was used for both experiments, and a wrist-mounted force/torque sensor was used to get validation data.

Both experiments were carried out twice, the first time with the nominal servo controller parameters, and the second time with detuned servo controllers, with $K_i = 0.03K_i^{nom}$, as was used in Sec. 8.2. The parameters for the joint control error method, i.e., the joint stiffnesses, were estimated from the same type of experiments that was described in Sec. 8.2. The identification experiments were carried out for each servo controller setting. The parameters for the motor torque method were estimated from experiments that were described in Sec. 8.3. In contrast to the friction model parameters chosen in Sec. 8.3, the B-parameter in (8.8) was chosen to be larger such that a larger velocity was required to reduce the uncertainty than the parameter choice displayed in Fig. 8.11. This means that the velocity noise influence will be reduced when the joints are not moving. The assumption of small contact torques were used for both estimation methods, i.e., choosing R_F as was done in Secs. 8.2 and 8.3.

Applying forces to a static robot

In the first experiment, forces were manually applied to the end effector of the robot. The configuration of the robot together with the coordinate frame the forces were measured in is displayed in Fig. 8.19. The results from when the nominal parameters for the servo controllers were used are displayed in Fig. 8.20 (the motor torque method) and Fig. 8.21 (the joint control error method). The first column shows the result when forces were applied in the x-direction, the second column when forces were applied in the y-direction, and the third column when forces were applied in the z-direction. The approximate confidence intervals are also displayed for the motor torque method. It can be seen in Fig. 8.20 that the motor torque method detects all the applied forces with confidence, and the estimation error is relatively small. There are some false estimates for the directions that the forces were not applied in, but the measured force stayed within the confidence interval estimates. It can further be seen that the different choice of friction model (larger B in (8.8)) compared to what was chosen in Sec. 8.3 lead to significantly less noise, cf. Fig. 8.15.

The joint control error method, with results shown in Fig. 8.21, also detected the applied forces, but with quite large errors in magnitude, except for the y-direction. As was described in Sec. 8.2, the force estimates will be a high-pass filtered version of the actual applied forces. The forces

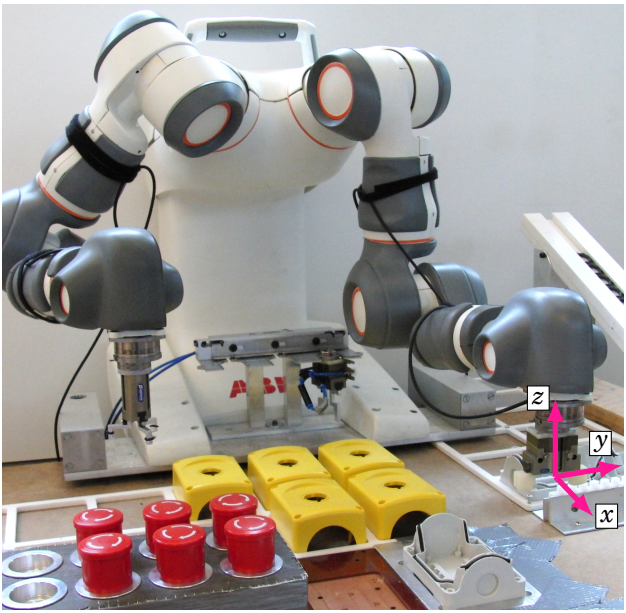


Figure 8.19 The setup for the experiments performed to compare the force estimation methods. The left arm of YuMi is in the configuration used for the experiment where forces were manually applied, and the coordinate frame the forces are measured in is displayed. The fixture for the assembly task can be seen in the lower right part of the photo, here with a bottom box placed inside it.

were, however, not static, and high-pass filtering will give them approximately the same appearance. There are some erroneous estimates in the directions that the forces were not applied in, e.g., the z-direction for the applied x-force, but they were quite small.

Similar experiments were performed with detuned servo controllers. The results are displayed in Fig. 8.22 for the motor torque method, and Fig. 8.23 for the joint control error method. The motor torque method performed just as well, or even better, as compared to when the nominal servo controller parameters were used. Especially the applied x-force was estimated with more confidence, i.e., tighter confidence intervals. The reason for this was that with detuned controllers, the applied force lead to a larger deviation of the robot and also a longer time period until the deviation was eliminated, i.e., the joints were moving and this gave a reduced Coulomb friction uncertainty. The joint control error method detected all the applied forces, but with large errors in magnitude, and also large false force estimates in the directions where the forces were

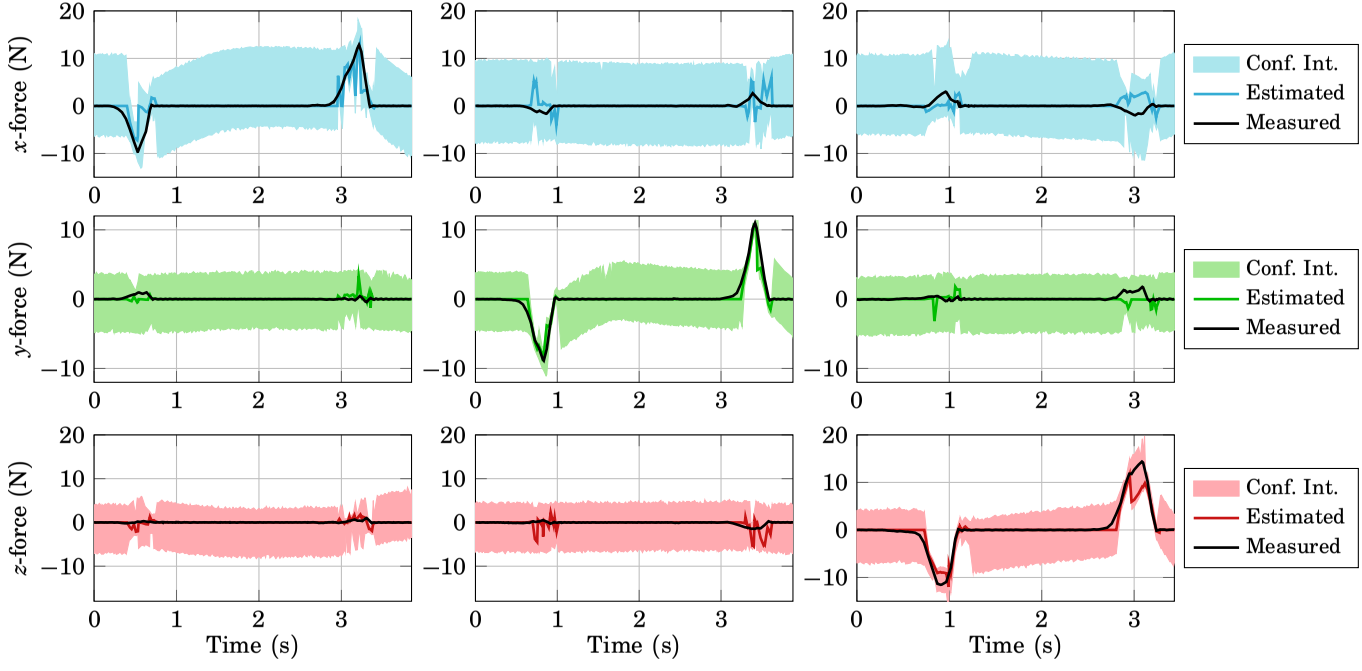


Figure 8.20 Measured and estimated forces from an experiment where forces were manually applied to the robot. The nominal servo controller parameters were used, and the motor torque method was used for estimating the forces. Each column corresponds to a separate experiment.

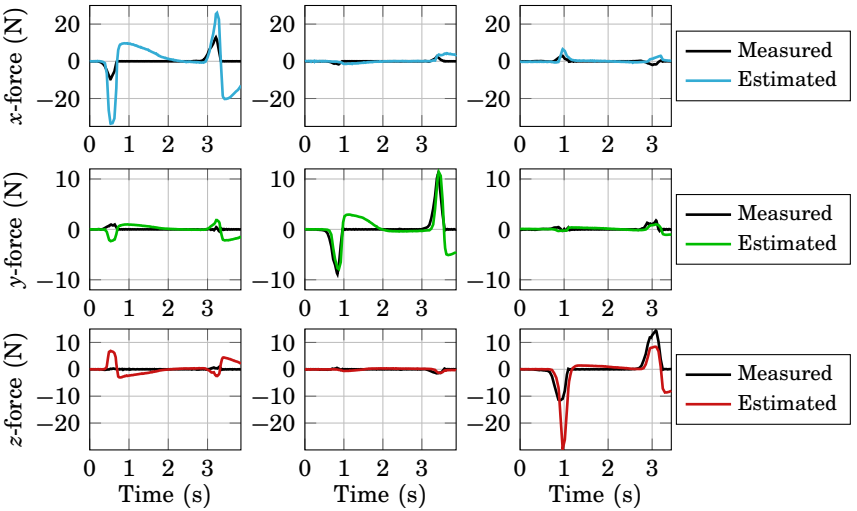


Figure 8.21 Measured and estimated forces from an experiment where forces were manually applied to the robot. The nominal servo controller parameters were used, and the joint control error method was used for estimating the forces. Each column corresponds to a separate experiment.

not applied. The detuned controllers were slow to eliminate the control errors, and this lead to false force estimates after each applied force.

In this experiment, the motor torque method performed best, irrespectively if the servo controllers were detuned or not. The joint control error method worked quite well when the nominal servo controller parameters were used, but the method could not be trusted if it was important to estimate correct force magnitudes.

Assembly task

The assembly task considered was to put the gray bottom box in the fixture in the emergency stop button assembly scenario, as described in Sec. 3.3. As none of the force estimation methods are suitable for controlling small static forces, the same strategy as for putting the yellow box with the button in the intermediate storage was used (Sec. 3.3), i.e., when a contact has been detected, keeping the position instead of controlling the contact force. The robot moves back 0.5 mm after each detected contact, to account for compliance in the gripper and the robot. An extra z-search was added in the end of the sequence, to make sure that the box was released in the fixture when the gripper opened. When the force sensor was used for the assembly, force control in the z-direction made

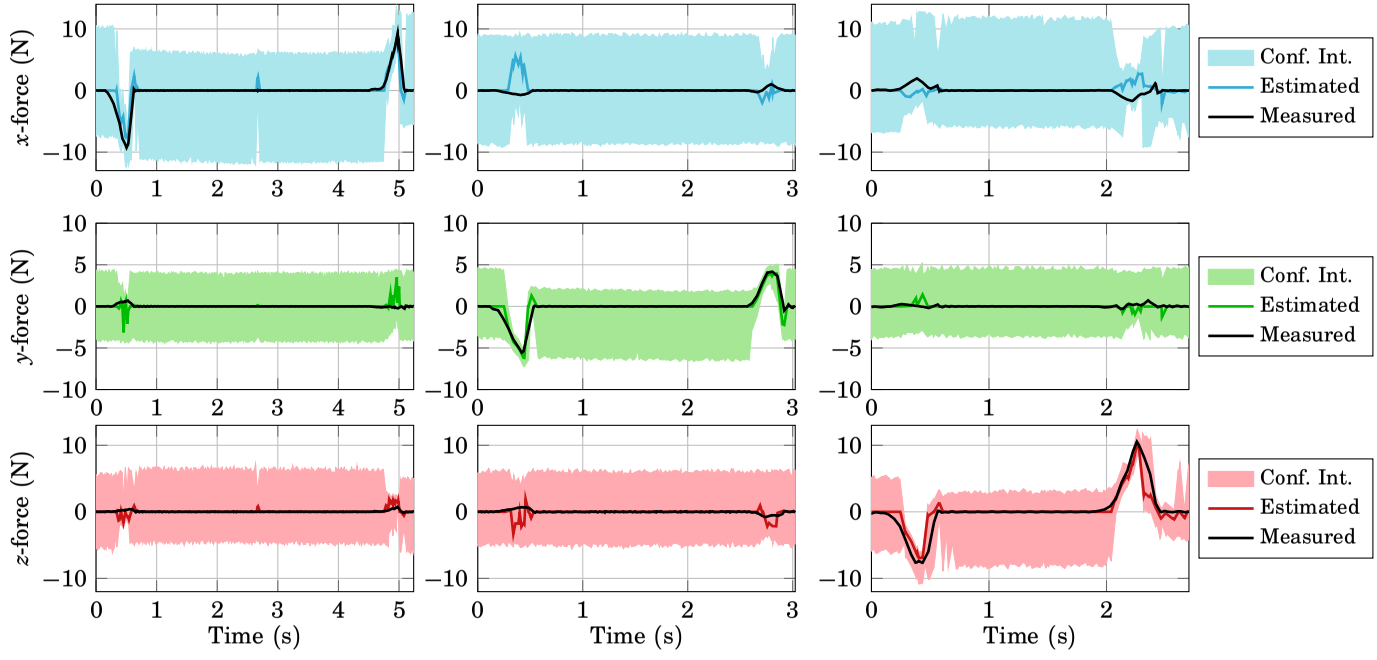


Figure 8.22 Measured and estimated forces from an experiment where forces were manually applied to the robot. The detuned servo controller parameters were used, and the motor torque method was used for estimating the forces. Each column corresponds to a separate experiment.

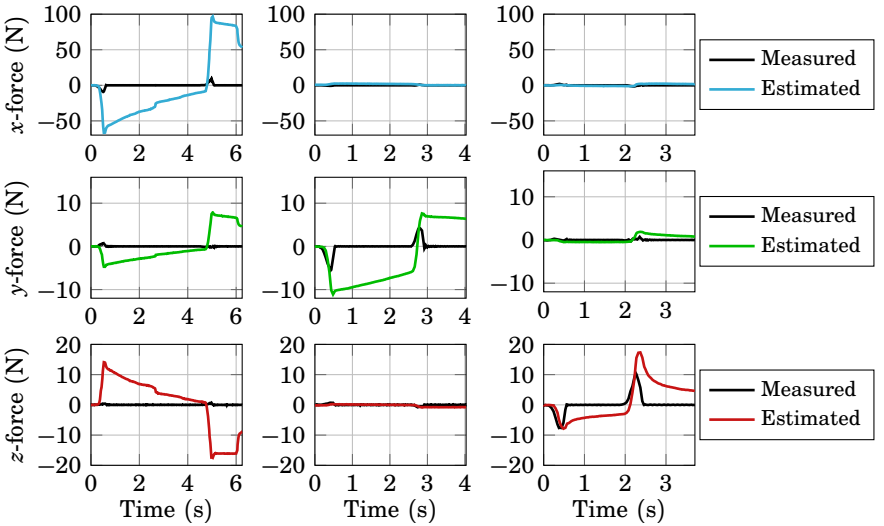


Figure 8.23 Measured and estimated forces from an experiment where forces were manually applied to the robot. The detuned servo controller parameters were used, and the joint control error method was used for estimating the forces. Each column corresponds to a separate experiment.

sure that the box was in contact with the bottom of the fixture.

The assembly task was first performed using the nominal servo controller parameters, and the motor torque method was used for detecting the contact forces. The estimated forces using the motor torque method are displayed in Fig. 8.24, and the forces estimated with the joint control error method in Fig. 8.25. The robot starts the first z-search from rest, which explains the large confidence intervals in the beginning of Fig. 8.24. The first force detected was the z-force at $t = 0.4\,\text{s}$, which was followed by a detected y-force at $t = 0.9\,\text{s}$. The next force to be detected was a large negative x-force, which appeared at $t = 1.4\,\text{s}$. The estimator, however, underestimated the force magnitude, but the force could still be detected. Finally, the last z-search was detected at $t = 1.7\,\text{s}$. The force estimator performed well in the beginning of the assembly task, but the performance was quite poor in the end. The relevant forces could, however, be detected with good robustness.

The estimated forces with the joint control error method are displayed in Fig. 8.25 (the same assembly execution that was displayed in Fig. 8.24). It can be seen that the force estimate gives the correct reaction for all the interesting forces, but there are also a lot of false estimates. Most of

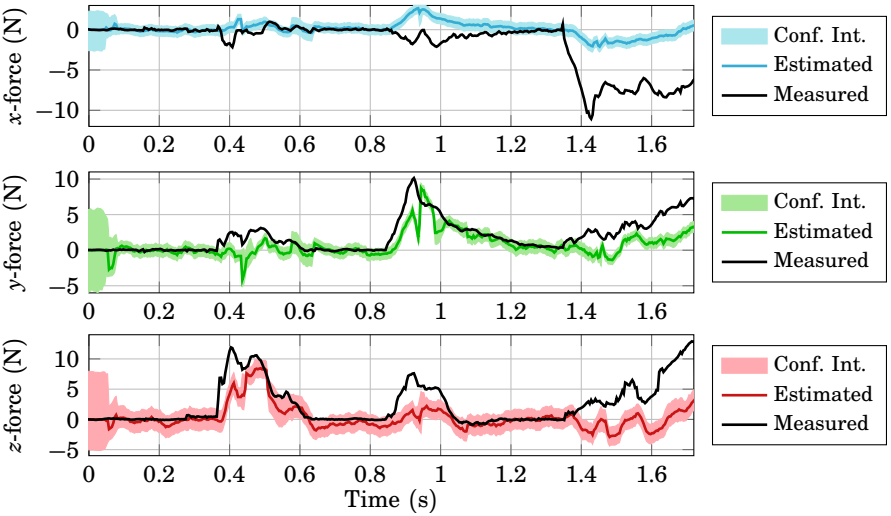


Figure 8.24 Measured and estimated forces from the assembly task, with nominal servo controller parameters and the motor torque method used for both force estimation and accomplishing the assembly.

the false estimates, however, occur for directions that are not interesting for the current state in the assembly sequence. The force estimate could therefore be used anyway, at least for the x- and the y-direction. For the z-direction, however, it would be difficult to use a threshold detector for the detections to be made at $t = 0.4\,\text{s}$ and $t = 1.7\,\text{s}$.

The assembly task was performed once again, but now with detuned servo controllers. Experimental data from an execution where the motor torque method was used for force estimation is displayed in Fig. 8.26. The result is very similar to the execution displayed in Fig. 8.24, and detuning of the servo controllers does hence not seem to have any effect on the force estimates.

Finally, the assembly task was performed one time where the joint control error method was used for detecting the contact forces (with detuned servo controllers), and the resulting estimated forces are displayed in Fig. 8.27. The first thing to notice is that the assembly time here is almost doubled compared to the executions in Figs. 8.24–8.26. The reason for this was that the assembly strategy had to be modified. In the previous executions, when a contact force was detected, the robot immediately started the next search motion while a position controller was initiated in the earlier search direction, with a position reference 0.5 mm retracted

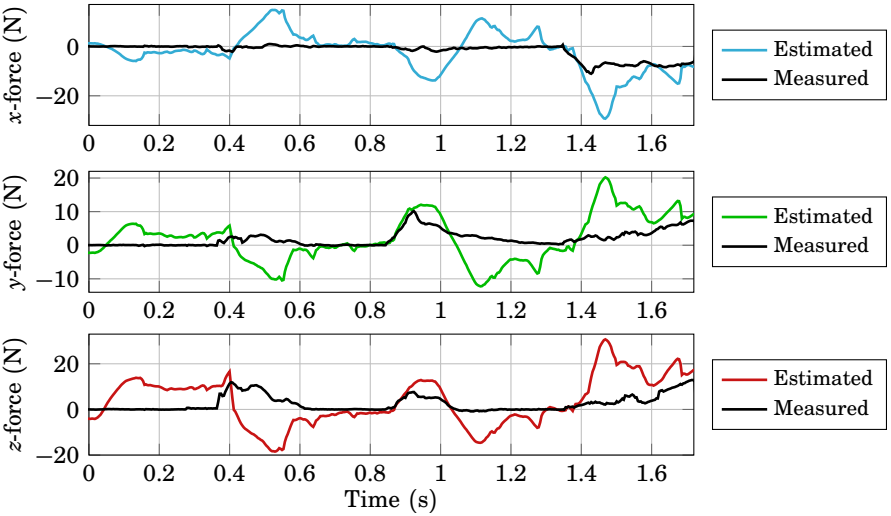


Figure 8.25 Measured and estimated forces from the assembly task, with nominal servo controller parameters and the joint control error method used for force estimation, but the assembly was accomplished using forces estimated with the motor torque method (the same execution as is displayed in Fig. 8.24).

from the detected contact position. The retracting movement gave, however, too large estimation disturbances for the joint control error method, i.e., the movements generated control errors that misled the estimator to believe that there were contact forces present. The remedy to this problem was to finish the retracting motion before the next search motion was started, and this was the reason for the increased assembly time. The force estimate in Fig. 8.27 can be seen to give the correct reaction for all the interesting forces, i.e., the z-force at $t = 0.4\,\mathrm{s}$, the y-force at $t = 1.3\,\mathrm{s}$, the x-force at $t = 1.9\,\mathrm{s}$, and the z-force at $t = 2.8\,\mathrm{s}$. The estimation errors in between the interesting forces were, however, very large, and it would be somewhat difficult to interpret the result without the validation force data.

Discussion

The joint control error method is a simple and approximate force estimation method. It has a simple calibration procedure and only requires access to the joint control errors. The estimate is calculated via a few matrix calculations, making it suitable for real-time implementation. The

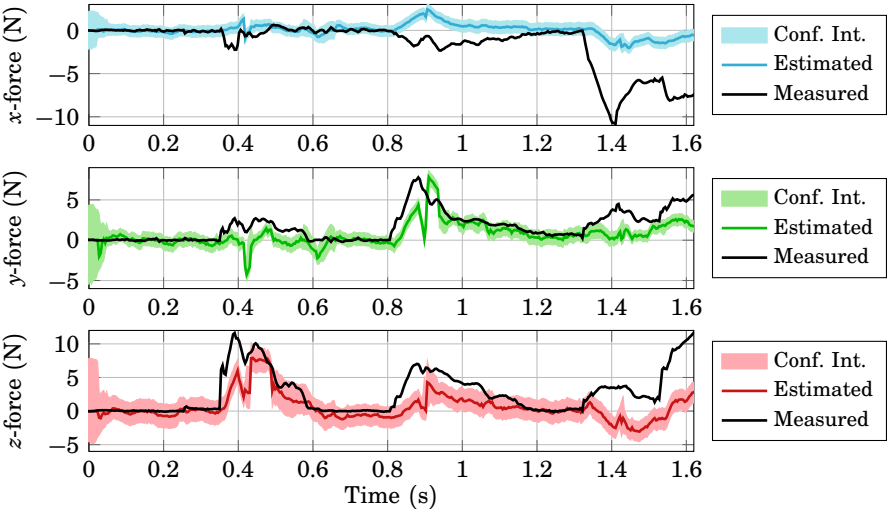


Figure 8.26 Measured and estimated forces from the assembly task, with detuned servo controller parameters and the motor torque method used for both force estimation and accomplishing the assembly.

main drawbacks are that it only gives high-pass filtered force estimates, that it gives large estimation errors which sometimes are hard to distinguish from the actual forces, and that it may require the servo controllers to be detuned.

The motor torque method is a theoretically justified force estimation method. It requires access to the motor torques, and the calibration procedure is more comprehensive than for the joint control error method, including gravity compensation and joint friction models. The force estimate is calculated by solving an optimization problem numerically. The method further provides approximate confidence intervals, which were shown to be trustworthy in experiments. A drawback with the method is the computation time required. A sampling time of 4 ms was used in the experiments, and this is about as fast the method can run on a standard computer to be able to both compute the force estimate and the confidence interval. The computation time can be sped up, however, by selectively choosing in which directions the confidence interval needs to be calculated. A drawback compared to the joint control error method is the need for a gravity compensation model.

Detuning the servo controllers does not make that much of a difference for the motor torque method. For the joint control error method, detun-

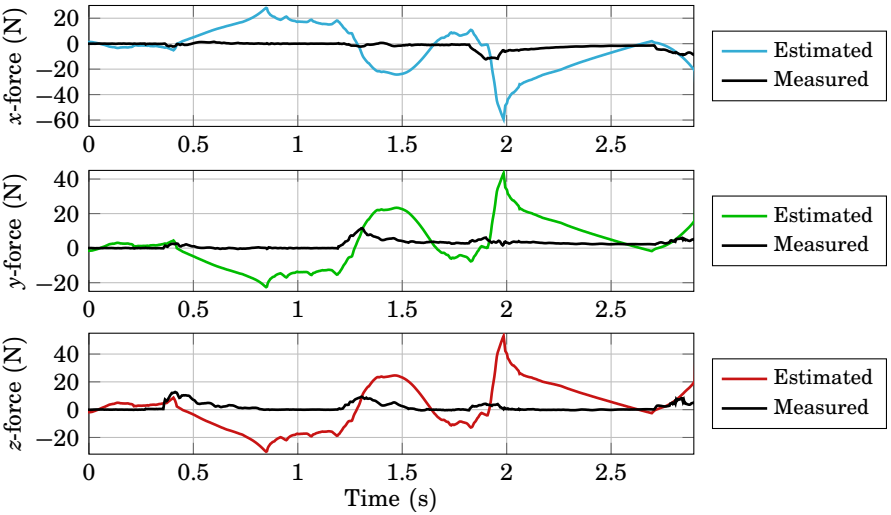


Figure 8.27 Measured and estimated forces from the assembly task, with detuned servo controller parameters and the joint control error method used for both force estimation and accomplishing the assembly.

ing was not beneficial in the experiment where forces were applied to a static robot, as was seen in Figs. 8.21 and 8.23. But in the assembly task, the robot became too stiff without detuning, such that it was hard to distinguish control errors coming from the motion of the robot from control errors due to external forces. Detuning the servo controllers makes the robot more compliant, and this is something that can be beneficial in assembly tasks. This effect was, however, not especially visible in the performed experiments. An advantage with the motor torque method is that the detuning is optional, while the joint control error method sometimes requires it, such as in the assembly task.

Both methods were shown to be able to estimate forces in several tasks. The motor torque method is superior when it comes to giving accurate estimates, and giving confidence bounds. The joint control error method is, however, much simpler to implement and calibrate, and it can therefore be a serious alternative in some tasks, although it might require some more time for designing the assembly strategy to use, e.g., as in the performed assembly experiment where extra states had to be added to finish the retracting movements to avoid too large erroneous force estimates.

8.5 Force estimation for industrial robots

All experiments presented so far in this chapter were performed with YuMi. This is a light-weight robot with quite low friction torques. In this section, force estimation with the IRB140 is considered. It is a much heavier and stiffer robot than YuMi. The friction torques are approximately a factor of ten larger for IRB140 than for YuMi, also the noise levels are a factor of 5–10 larger for IRB140. The larger friction and noise levels do not, however, depend on the gear ratio, as they are approximately the same as for YuMi. In addition to the larger disturbances, the IRB140 contains one joint less, i.e., one less measurement to use. These properties make it difficult to estimate small external forces.

The force estimation method based on the motor torques presented in Sec. 8.3 will be used. The estimation performance is evaluated in two experiments. First, forces were manually applied to a static robot, and secondly, the snapfit assembly task described in Sec. 3.2 was accomplished.

Extended friction model

The YuMi-robot is constructed such that there is one separate gearbox for each motor. The IRB140, however, has a coupled gearbox for the three wrist joints. This means, e.g., that moving the fourth motor will result in that all three wrist joints will move on the arm side. Another effect is that although a motor is moving with a constant velocity, the arm side of the joint may have a non-constant velocity depending on how the coupled motors are moving; the arm side of the joint may, for instance, change the direction of movement. To account for these effects, the friction on both the arm side and the motor side have to be modeled. The result from one experiment where the fourth and the fifth motor have been moving with piecewise constant acceleration is displayed in Fig. 8.28, which shows the motor torque for the fifth joint. Here it can be seen that there are both arm side and motor side Coulomb friction present, thus confirming the need for the extended friction model.

The total friction torque, τ_f, experienced by the motors will now be modeled as

$$\tau_f = \underbrace{\tau_{C,motor} + \tau_{visc,motor}}_{\tau_{f,motor}} + J_{gearratio}^T \left(\underbrace{\tau_{C,arm} + \tau_{visc,arm}}_{\tau_{f,arm}} \right) \tag{8.22}$$

where τ_C denotes Coulomb friction, τ_{visc} denotes viscous friction, and where limits on each component of $\tau_{f,motor}$ and $\tau_{f,arm}$ are given by models on the form (8.8). Further, $J_{gearratio}$ is the gear-ratio matrix, relating how the arm side of the robot will move when the motors move, i.e.,

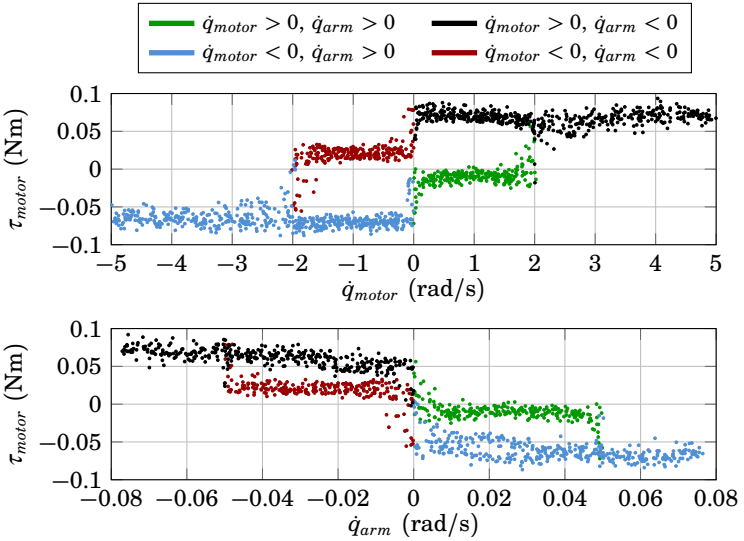


Figure 8.28 The motor torque for the fifth motor of the ABB IRB140 versus the motor velocity (upper diagram) and the arm side velocity (lower diagram). Apart from the fifth motor, the fourth motor was also running, which is the reason for that the arm side velocity changes direction. It can be seen that Coulomb friction exists both on the motor side and the arm side.

$\dot{q}_{arm} = J_{gearratio}\dot{q}_{motor}$. The relation (8.22) can be rewritten as

$$\tau_f = \underbrace{\begin{bmatrix} I & J_{gearratio}^T \end{bmatrix}}_{J_{fric}^T} \underbrace{\begin{bmatrix} \tau_{f,motor} \\ \tau_{f,arm} \end{bmatrix}}_{\tau_{f,tot}} \tag{8.23}$$

For the ABB IRB140, the gear-ratio matrix has the following structure

$$J_{gearratio}^T = \begin{bmatrix} * & 0 & 0 & 0 & 0 & 0 \\ 0 & * & 0 & 0 & 0 & 0 \\ 0 & 0 & * & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & * \end{bmatrix} \tag{8.24}$$

where stars ($*$) indicate non-zero elements. As can be seen, only arm side friction for the fifth and the sixth joints will influence some other motor than the corresponding one, i.e., arm side friction for the fifth joint will influence also the fourth motor apart from the fifth motor, and arm side

friction for the sixth joint will also influence the fourth and the fifth motor apart from the sixth motor. It is thus only necessary to model the arm side friction for the fifth and the sixth joints. For all other joints, it will be impossible to separate the arm side friction from the motor side friction, assuming that the coupling between the motor and the arm side of the joint is rigid. All friction for the first four joints are therefore modeled to be motor side friction. The optimization problem to solve to get the force estimate is now given by (cf. (8.13))

$$\begin{aligned} \underset{\text{over } F,\tau_{f,tot}}{\text{minimize}} \quad & \tfrac{1}{2}\left(\bar{\tau} - J_m^T F - J_{fric}^T \tau_{f,tot}\right)^T R_e^{-1}\left(\bar{\tau} - J_m^T F - J_{fric}^T \tau_{f,tot}\right) \\ & + \tfrac{1}{2}\left(F - \bar{F}\right)^T R_F^{-1}\left(F - \bar{F}\right) \\ \text{subject to} \quad & \tau_{f,min} \leq \tau_{f,tot} \leq \tau_{f,max} \end{aligned} \tag{8.25}$$

where $J_m = JJ_{gearratio}$ and J is the robot Jacobian such that the arm side joint torque $\tau_{arm} = J^T F$, and the corresponding relationship for the motor side joint torque is $\tau_{motor} = J_m^T F$. Further, J_{fric} is a 8×6-matrix and $\tau_{f,tot}$ is a vector with 8 elements. Note that all equations in Sec. 8.3 were expressed on the arm side, i.e., the τ_f in Sec. 8.3 was the friction torques on the arm side, while τ_f in this section is the friction torques on the motor side.

Experimental results

Calibration The parameters for calculating the gravity torque, τ_{grav}, were identified from an experiment where the robot was slowly moving around in its workspace. The identified parameters resulted in a mean absolute error ranging from 0.5 Nm for the wrist joints to 2 Nm for the base joints. To tune the friction model parameters (in the model (8.8)), special care had to be taken when designing the experiments as some of the off-diagonal elements in $J_{gearratio}$ were small, such that it was possible to identify both the motor side and the arm side friction.

Forces applied to a static robot An experiment was performed where forces were manually applied to the end effector of a static robot, i.e., a robot controlled not to move. The result can be seen in Fig. 8.29, where it should be noted that each column of subplots shows the same experiment. It can be seen that large forces were required to get any reaction in the estimated force. The Coulomb friction for the base joints were 15–20 Nm, i.e., corresponding to forces of 30–40 N applied at the end effector with a lever arm of 0.5 m. This explains why the applied forces in this experiment hardly gave any reactions in the estimated force, as either the applied forces gave no reaction in the motor torques, or the reaction could be explained as friction torques. The good performance in the y-direction (relative to the other directions) can be explained by a favorable

configuration of the robot. In this direction, the applied forces had a quite long lever arm to the fourth joint, which like the other wrist joints, had comparably low Coulomb friction.

Assembly task The snapfit assembly scenario, described in Chap. 3, was used as an example assembly task. This task is challenging to perform using force estimation, considering that quite small forces need to be detected in order to accomplish the assembly task without damaging the parts involved. The original assembly strategy, illustrated in Figs. 3.4 and 3.5, was based on that contacts were force controlled after they had been established. This would be difficult using force estimation, due to the relatively small contact forces and the large uncertainties in the force estimate. An alternative strategy is to use position control after a contact has been detected, which will work if the orientation of the contact surfaces the robot searches along are known with sufficient accuracy. In the scenario considered, the bottom box was placed in a fixture, and the orientation of the box should thus be known with the required accuracy. The original strategy further uses some contact torque threshold levels to transition between the search motions. Doing this with force estimation will be difficult, as large torques will be required to make it possible to detect them, with risk of damaging the parts. The torques to be detected, however, originate from two-point contacts (as was earlier exploited in Chap. 7), and they will therefore also give rise to forces. The ψ-torques can therefore be detected by large z-forces, but the φ-torque threshold was not possible to replace with an x-force threshold, as no net x-force could be detected in this state. The remedy was to abandon the small torque assumption for the φ-torque, which was not valid anyway, but keeping it still improved the force estimates. Further, large contact torques had to be accepted to make it possible to detect the torque.

Measured and estimated forces from an execution of the assembly task are displayed in Fig. 8.30 and the torques in Fig. 8.31. It can first be noted that the confidence intervals are significantly tighter now, which is due to the fact that most joints were moving, which means that the Coulomb friction torques were assumed to be known and there were thus significantly less uncertainties left compared to when the robot was static. The force estimate was far from perfect, but the measured force was within the confidence interval estimates most of the time. To be certain that a force was present, both the estimated force and the confidence interval were considered, e.g., at $t = 1\,\mathrm{s}$, a large z-force was detected as both the estimated z-force was above $10\,\mathrm{N}$ and the lower confidence interval limit was larger than $5\,\mathrm{N}$. The estimator seemed to have problems to separate x- and z-forces, e.g., around $t = 4.5\,\mathrm{s}$ there is both an x- and a z-force present, but according to the estimator, both forces were most likely zero.

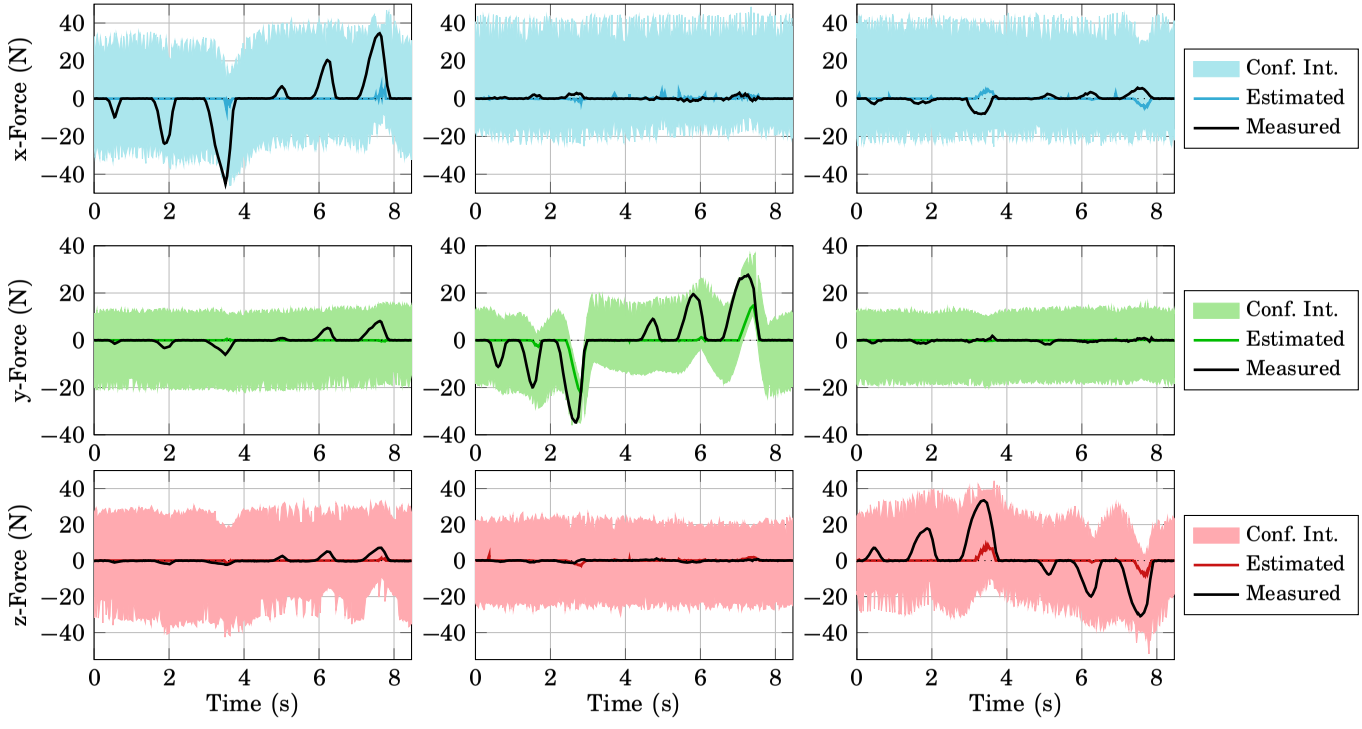


Figure 8.29 Measured and estimated forces from an experiment where forces were manually applied to the end effector of the ABB IRB140. Note that each column of subplots corresponds to separate experiments.

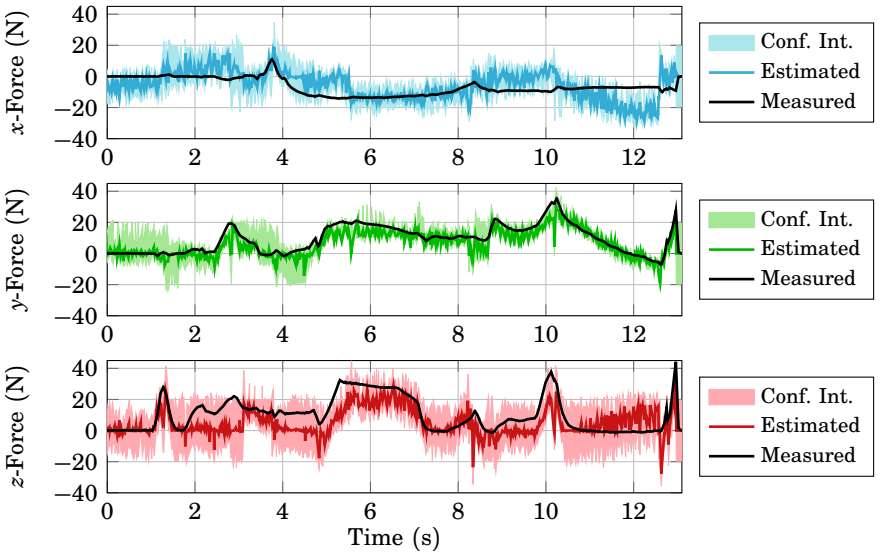


Figure 8.30 Measured and estimated contact forces from the execution of the snapfit assembly scenario using the IRB140.

This effect was probably due to the fact that it was the same joints that were beneficial for estimating both of these forces.

The most difficult part of the assembly task using force estimation was to perform the two last guarded search motions, i.e., state number 6 and 7 in Fig. 3.4. The transition for state number 6 was detected as a large z-force at $t = 5.3\,\mathrm{s}$. In the ψ-torque diagram (Fig. 8.31), this can be seen as a large ψ-torque, which the estimator does not detect. The assumption of small contact torque is not valid here, and this is probably the reason why the z-force was underestimated. Changing the assumption, however, did not lead to that the torque could be detected, and it further led to that the overall estimation performance was degraded. For state number 7, the guarded search in the φ-direction, the transition condition used with a force sensor was that the φ-torque exceeded 0.25 Nm. With force estimation, the torque had to exceed 1.5 Nm before the estimator was certain that it appeared, see the top diagram in Fig. 8.31 at $t = 8.2\,\mathrm{s}$. The electric switch was ”quite strong” for applied torques in this direction and could therefore handle it. It can also be noted that the noise level for the estimated φ-torque was significantly larger than for the other torque-directions. This was due to the larger variance of the prior distribution; the estimator had more freedom to explain the noisy measured motor

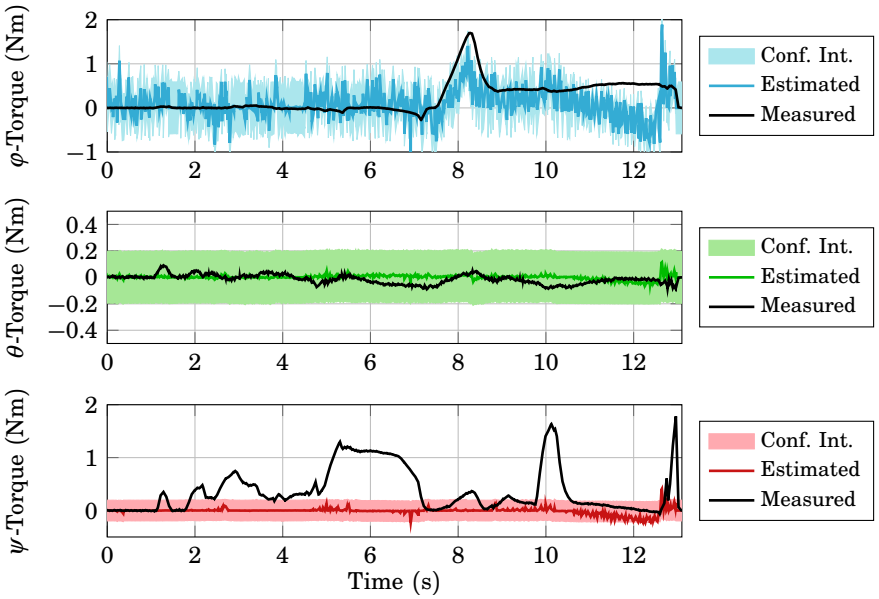


Figure 8.31 Measured and estimated contact torques from the execution of the snapfit assembly scenario using the IRB140.

torques as contact torque in the φ-direction. Further, in this state (number 7), the switch slides into the slot, and to make sure that this actually happened, both the y- and the ψ-position references were changed if the estimated y- and z-forces either became high or low.

The remaining operation was to snap the switch into the slot. This was performed in three steps. First a search in the ψ-direction was performed. During this search, it was made sure that the switch was pressed into the slot by requiring a large y-force, but the switch would not snap into the final position if it was pushed too hard into the slot. Therefore, after the first ψ-search had triggered (at $t = 10\,\text{s}$), the y-position reference was changed such that no net y-force was applied (the reference was changed as long as there was an estimated y-force present), and then one more ψ-search was performed, which ended when a large z-force was detected at $t = 13\,\text{s}$. The described procedure usually resulted in that the switch became correctly assembled.

Discussion

The experiment where forces were applied to a static robot showed that the force estimator required quite large forces to be certain that there was an external force present. The reason was the large friction levels in the IRB140, whereas the corresponding experiment with YuMi, Figs. 8.15 and 8.20, showed much better responsiveness.

Accomplishing the snapfit assembly task with IRB140 using force estimation was difficult, as quite small forces had to be detected and as the assembly strategy had to be carefully designed not to damage the involved parts. For this particular task, the accuracy of force estimation was barely enough, but it showed the potential of using it. For force estimation to be a serious alternative, larger forces/torques should be allowed to make it possible to make robust detection. An alternative is to use a robot with less disturbances, e.g., YuMi instead of the IRB140.

8.6 Discussion

In this chapter it was shown that force estimation could be used to replace a force sensor in numerous assembly tasks. The accuracy of the estimator is not as good as a force sensor, but usually good enough to be able to detect the forces of interest. A drawback with using force estimation is that it is usually not possible to use the same assembly strategies as with a force sensor, due to problems with estimating static forces and contact torques. Some of the benefits are that no investment in an expensive sensor is needed, and nothing has to be mechanically mounted on the robot that, for instance, decreases the payload.

The use of a prior distribution for the external force F with small variance on the contact torques can be seen as putting a soft constraint on the contact torques. Instead of estimating a three degrees of freedom force and a three degrees of freedom torque, the problem is then almost reduced to estimating only a three degrees of freedom force. The increased redundancy in the problem gives a better force estimate.

The prior distribution for F was chosen to be Gaussian, which is a really crude approximation of the true distribution. Using a Gaussian prior, however, leads to fast calculations, and it has been shown that it can also give a significant performance increase, despite its simplicity. The variance of the prior should be chosen according to process knowledge; how large forces and torques that are expected.

The current trend in robotics, where the robots are becoming more light-weight and safe to use in proximity to humans, is good for force estimation. These new robots usually have lower friction levels than traditional industrial robots, which makes it possible to achieve good force

estimation performance. The experiments in this chapter clearly showed that force estimation worked better for YuMi than for the traditional industrial robot IRB140.

Other methods for force estimation commonly rely upon that a dynamical model of the robot is known, e.g., [Alcocer et al., 2003; Eom et al., 1998; Van Damme et al., 2011]. Such dynamical models were not available for the robots used in experiments in this thesis and, therefore, these methods were not tested.

8.7 Conclusions

Two different methods for estimating forces without any force sensor have been presented. The first method was based on viewing the controlled robot joints as virtual springs, and approximating the joint position control errors as torques applied to the joints. The second method was based on the motor torques and modeling of the velocity dependent friction uncertainties. Both of the methods were subject to large disturbances, but the resulting force estimates could anyhow be used to replace a force sensor in many tasks. This was exemplified in implementations of multiple assembly tasks with different robots.

The motor torque method was better than the joint control error method at estimating the correct force magnitude, but both methods could be used to detect force transients. Neither method was good at estimating small static forces, and the assembly strategy where established contacts were force controlled had to be modified.

In contrast to most previously published methods for force estimation, the methods presented in this chapter were implemented on real industrial robot systems and successfully applied in real world assembly tasks.

9

Dithering to Improve the Accuracy of Force Estimation

9.1 Introduction

Force sensing can give robots the capability to correct for small position uncertainties by sensing the contact forces. This makes it possible to accomplish, for instance, assembly tasks in a robust way as was presented in Chap. 3. A force sensor may also be used for easy programming of a robot. By letting the robot be manually guided by an operator, a lead-through programming scenario, the robot can easily be taught what to do. A force sensor may, however, be too expensive, especially if lead-through programming is the only intended use. A sensor may also be sensitive to different environments, e.g., varying temperatures, and it may add unnecessary mass to the system, i.e., reducing the effective workload. An alternative is therefore to estimate the external forces, by using sensing already available in the robot, see further Chap. 8.

When a robot is not moving, the main disturbance for force estimation is the Coulomb friction. It is usually quite large for industrial robots, which often have gear boxes with high gear ratios. This means that large external forces are needed to overcome the friction and make it possible to estimate the force, as was seen in Sec. 8.5. When the robot is not moving, the friction force may be anywhere within the friction band, and the force to overcome the friction is thus unknown. Previous research has been performed about compensating for friction, e.g., in [Freidovich et al., 2010] the dynamic LuGre model was used for friction compensation. For force estimation, however, knowing the friction in more detail will not give better estimates, as the external force still must overcome it. By using a feedforward torque within the friction band, however, the torque can be

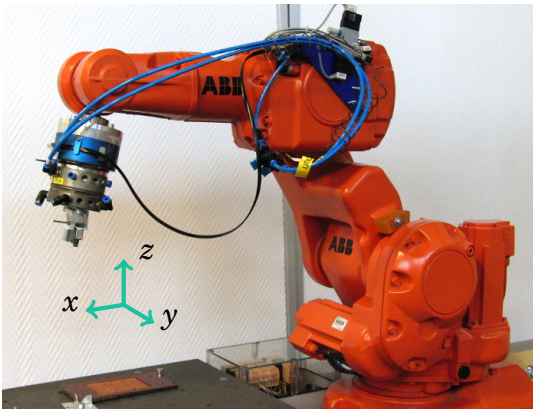


Figure 9.1 The ABB IRB140 robot used in the experiments. The wrist-mounted JR3 force/torque sensor (blue cylinder) was used for verification of the estimated entities. The reference coordinate frame is also displayed.

controlled to be close to the border of the friction band. Then, only a small external force is needed to overcome the friction. This feedforward torque may be in the form of a dithering signal, i.e., a high-frequency zero-mean signal.

Dithering has previously been used, for instance, to suppress quantization effects. This has been done by adding the dithering signal as a noise signal. A survey of its use in audio signal processing is presented in [Lipshitz et al., 1992]. Another application where dithering was used to suppress quantization effects is precise current control [Zhu and Fujimoto, 2013]. Dithering has also been used in the context of robotics. In [Ipri and Asada, 1995; Lee and Asada, 1995] dithering was used in assembly tasks to overcome friction between the parts, which corrupted the force measurements. It was further presented how the dither parameters could be tuned online. Dithering can also be used to compensate for stiction in valves. In [Hägglund, 2002] a dithering-like signal was added to the control signal to compensate for stiction in pneumatic control valves in the process industry.

This chapter considers the problem of force estimation when the robot is not moving and subject to Coulomb friction. A dithering signal was used to decrease the static friction uncertainty, and hence increase the force estimation accuracy. The method was implemented and tested experimentally with an industrial robot in a lead-through programming scenario. The experimental setup is displayed in Fig. 9.1, where the fixed workspace reference directions are illustrated.

A similar approach to the one presented in this chapter where force estimation was performed together with dithering to suppress the friction uncertainties is presented in [H. Cho et al., 2014]. The external torque applied to each joint of the robot was estimated through disturbance observers, and a dithering signal was used to make the robot more sensitive to external forces. Another force estimation method using dithering is presented in [Chen and Kazanzides, 2013]. The method is targeted for non-backdrivable robots, in contrast to the method presented in this chapter. A sawtooth dithering signal was used to make the robot move slightly, as the non-backdrivability made it impossible to estimate forces when the robot was not moving.

9.2 Dithering

The robot used for experiments in this chapter was the IRB140, see Fig. 9.1. The interface available to the robot was described in Sec. 2.2.

One joint

To see the benefit for force estimation of using dithering, a simple experiment was first performed. A force was applied to the robot by manually pushing its end effector, and the resulting motor torque for one of the joints is shown in the upper diagram in Fig. 9.2. The diagram also contains the external torque, τ_{ext}, as measured by a force sensor, and an estimate of the Coulomb friction band. It can be seen that there is not much of a response in the motor torque data, although the last push exceeded the friction band. The joint controller was active in the experiment, controlling the position of the joint, otherwise the last push would have resulted in the robot starting to move.

The same experiment as above was conducted once more, but this time with a torque feedforward dithering signal, see the lower diagram in Fig. 9.2. The last two pushes are now clearly visible in the motor torque data, and almost the first one as well. By adding the dithering signal, it is thus possible to detect smaller forces than was possible without the dithering. This result is similar also for the other joints of the robot.

Dithering signal generation

The dithering signal used in this chapter was a square wave, and it was sent to the robot system as a torque feedforward signal, τ_{ffw} in the block diagram of the joint controller in Fig. 2.4. When the robot is not moving, the torque for each joint may be anywhere within the Coulomb friction band. Therefore, to center the dithering signal inside the friction band requires some control. A feedback control scheme was used in this case.

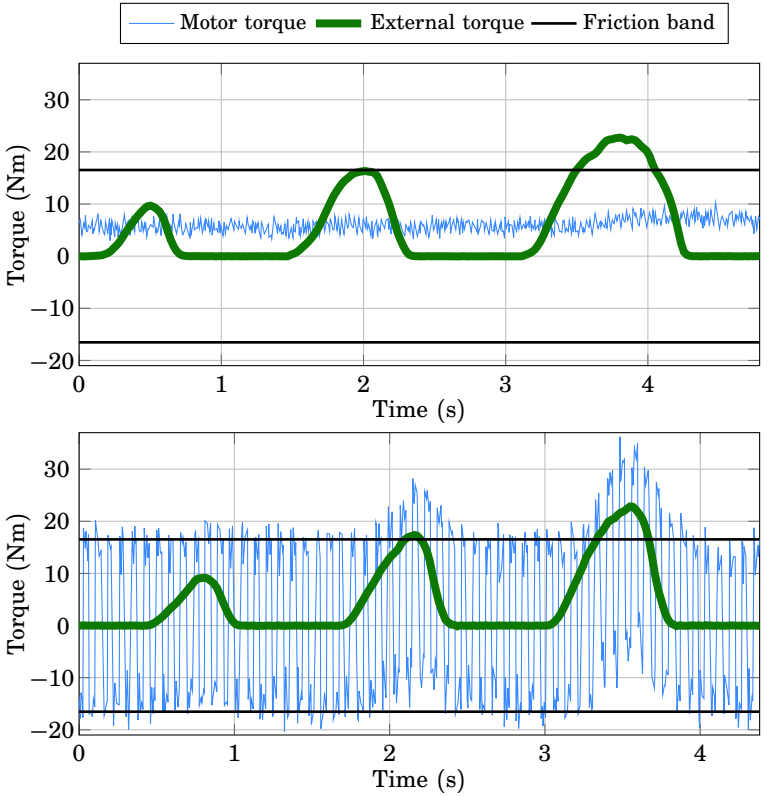


Figure 9.2 Resulting torques for one joint from an experiment where forces were manually applied to the robot, without dithering in the upper diagram and with dithering in the lower diagram.

To be able to control towards the center at all, the position loop in the joint controller had to be disabled, i.e., setting $K_p = 0$ in Fig. 2.4. Otherwise, the position control loop would counteract any static torque feedforward. One reason for this could be that the motor could move slightly without moving the arm side of the joint, with the gears acting as a spring for torques within the friction band. The reference for the center torque was in the middle of the estimated friction band, i.e., the estimated torque due to gravity. The center of the measured torque signal was calculated as the mean over an integer number of periods of the dithering signal, see illustration in Fig. 9.3. An integral controller was used for closing the feedback loop.

The amplitude of the dithering signal also had to be controlled. The

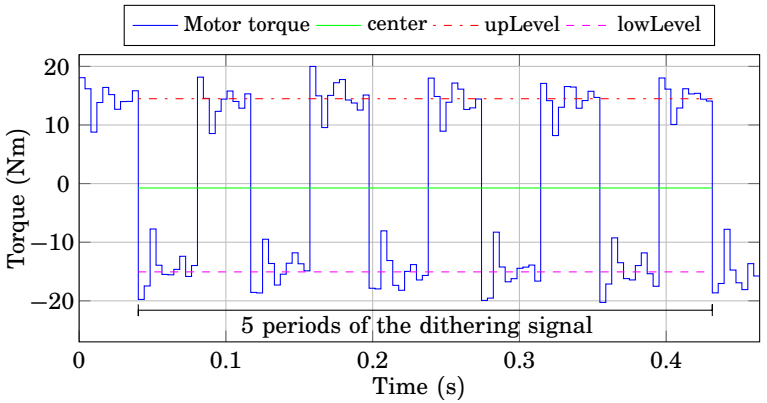


Figure 9.3 Illustration of how the dithering signal parameters are calculated. The center is calculated as the mean of an integer number of periods, five periods in the diagram. The upLevel and the lowLevel are the means of the samples above and below center, respectively. The amplitude is calculated as half the difference between upLevel and lowLevel. The calculated values are used at the end of the marked interval, i.e., at $t = 0.43$ s.

motor torque measurements were actually the references sent to the current control loops in the motors, as true motor torque could not be measured. This means that to be able to detect external torques, the joint controllers had to be active. However, this also resulted in the control counteracting the dithering signal, meaning that a slightly larger feedforward signal was needed than the desired torque response. A feedback loop with an integral controller solved this problem. The amplitude was calculated as the mean of half the difference of all samples above and below the center for an integer number of periods of the dithering signal. The pseudo-code below was used for the calculation:

```
function calcDitherSignalPars(trq)
    trqHistory ← [trq , trqHistory(1..end-1)]
    center ← mean(trqHistory)
    upLevel ← mean(trqHistory>center)
    lowLevel ← mean(trqHistory<center)
    amplitude ← (upLevel-lowLevel)/2
    return center, amplitude
end function
```

The variable trqHistory is a persistent variable between calls to the function, and it contains a history of the torque for an integer number of dithering periods. The calculation of upLevel (and lowLevel) is done by

taking the mean of all samples being greater than (lower than) the mean of trqHistory. An illustration is given in Fig. 9.3, where five periods of the dithering signal was used. The value for the calculated signals in the figure are used in the end of the marked interval, i.e., at $t = 0.43$ s.

When an external torque appears, the feedback controllers will try to counteract the deviations. But as the measurement signals are based on an average over the last periods, and as the control loops are not too tight, it will be possible to detect the external torque. Once it has been detected, the robot is supposed to act upon it and turn off the dithering signal.

The set-point amplitude for each joint was chosen as large as possible before the dithering signal resulted in a vibrating robot. About one hundredth of a motor radian, which would correspond to 0.0001 arm radians, was chosen as an acceptable level of vibration. The dithering amplitude for the different joints varied between 50–90 % of the estimated friction band.

9.3 Force estimation using dithering

The method for force estimation used in this chapter is the motor torque method presented in Secs. 8.3 and 8.5.

Extracting torque signal and friction bounds

As was seen in Fig. 9.2, when a joint was fed with a dithering signal, the external torque appeared as a superimposed signal on the dithering signal. Actually, the external torque appeared on both sides of the dithering signal. Another example of a dithered joint where an external torque has been applied is displayed in the upper diagram of Fig. 9.4. The two curves marked as upper (red) and lower torque (green) are the one-period means of all samples above and below the mean of all samples within the last dithering period, respectively. The mean of these two signals should be an estimate of the applied external torque. Taking the mean has the effect that it shifts the signal to the center of the dithering signal, as well as taking away some noise. The lower diagram of Fig. 9.4 shows the estimate together with the actual external torque, as measured by a force sensor for verification. Whereas the actual torque is underestimated, the shape of the signal is captured. A slight response delay can also be seen.

As the external torque is added to the dithered signal, only a small external torque is required to exceed the friction band. This has the effect that the friction band is effectively decreased. In Fig. 9.4 the set-point for the dithering signal was 90 % of the friction band, and thus only 10 % of the friction band remained. This is illustrated in the lower diagram in

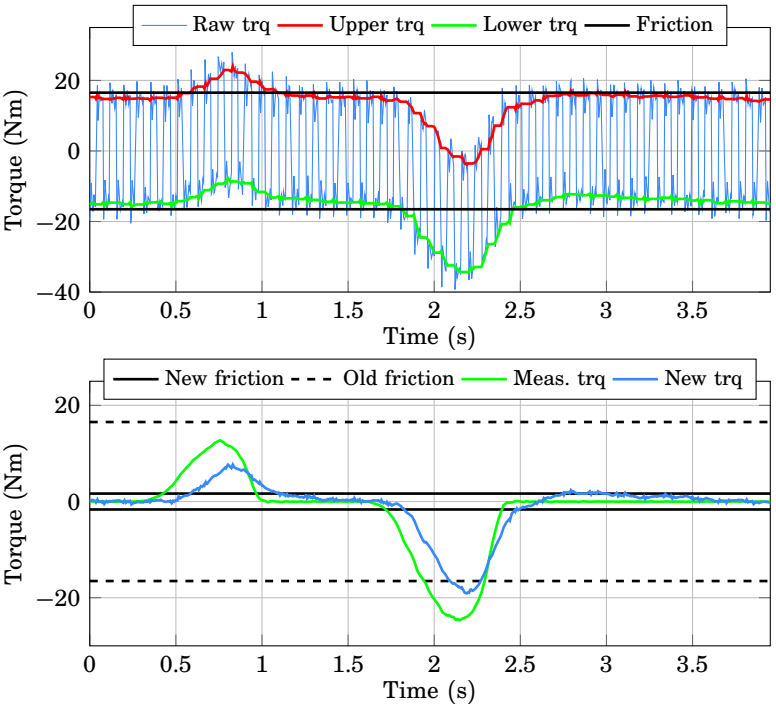


Figure 9.4 Illustration of how the torque signal and Coulomb friction bounds are extracted. The upper diagram displays how an upper and a lower torque signal are calculated. The lower diagram shows the new torque signal, the mean of the upper and lower torque in the upper diagram, together with the external torque, as measured with a force sensor. The new and old friction band estimates are also displayed.

Fig. 9.4, where the new friction band is the old friction band shifted down with the dithering set-point amplitude.

Dithering on several joints

When dithering signals are sent to all joints of the robot simultaneously, there might be interactions between the joints which amplifies the vibration. This meant that the dithering amplitude had to be decreased for some of the joints, as compared to when the joints were dithered separately.

9.4 Application scenario

The dithering method was implemented in a lead-through programming scenario, i.e., to let the user guide the robot by holding on to the end effector (or to any other part of the robot). For the force estimation, it was assumed that all external forces were applied at the end effector. This means that the torques around the end effector will be small, and this information was used for choosing the prior, $\bar{F} = 0_{6\times 1}$ and $R_F = \text{diag}(20\text{ (N)}, 20\text{ (N)}, 20\text{ (N)}, 0.3\text{ (Nm)}, 0.3\text{ (Nm)}, 0.3\text{ (Nm)})^2$ in (8.25), i.e., forces of 20 N expected and only small contact torques. Dithering was initiated when the robot was still, and it was turned off when a force was detected that made the robot move.

Only translational movements with a fix orientation were allowed, due to the difficulty of estimating external contact torques. Long lever arms made estimation of forces beneficial, while the low signal-to-noise ratio for torques makes it almost impossible to estimate them, at least when only small contact torques were expected, as in the lead-through programming scenario.

The lead-through programming was accomplished by using a force controller, which was implemented as an impedance controller in task space with zero stiffness. To avoid that noise and incorrect force estimates moved the robot, a deadzone of 10 N was used on the force estimate. Another option could have been to use the confidence interval estimate and avoid using a deadzone, by setting the force to zero if the interval contains zero, and otherwise use the limit with the lowest magnitude. The confidence interval width, however, decreases rapidly when the robot starts to move, which results in that the force driving the robot will either be zero or quite large. The resulting robot motion would then be quite jerky and thus be discomforting for the operator.

9.5 Experimental results

Robot system

The robot used in the experiments was an IRB140. A wrist-mounted JR3 force/torque sensor was used for verification of the estimated forces, see the blue cylinder at the wrist of the robot in Fig. 9.1. The external joint torques were calculated as $\tau_{ext} = J^T F_{sens}$, where J is the robot Jacobian and F_{sens} the force/torque measurement from the sensor.

The friction model parameters in (8.25), R_e, $\tau_{f,min}$, and $\tau_{f,max}$, were estimated from the same types of experiments as were used in Sec. 8.3. The noise terms acting on the different joints were assumed to be independent, such that R_e became diagonal.

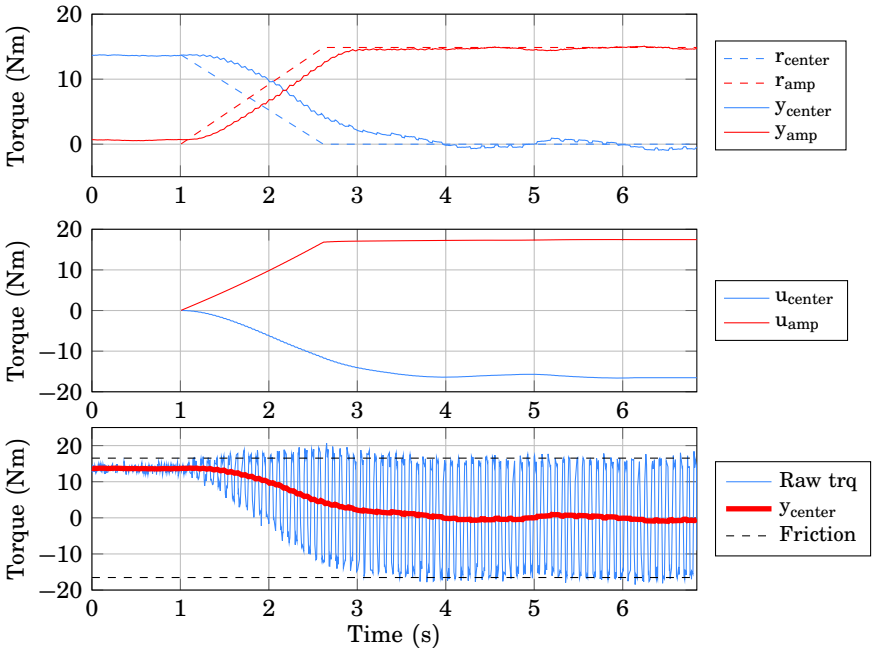


Figure 9.5 An example of the dithering signal control performance. The top diagram shows the reference and the measurement signals. The middle diagram displays the control signals. No reference or control signal are displayed until dithering is activated, which happens at $t = 1\,\text{s}$. The bottom diagram shows the actual motor torque signal together with the center torque measurement and the friction band estimate.

Control of dithering signal

Experimental data from one joint showing the control of the dithering signal are given in Fig. 9.5. The top diagram shows the reference signals and the corresponding measurement signals for the center level and the amplitude of the resulting motor torque signal. They were calculated by averaging over the five last periods of the dithering signal, as was illustrated in Fig. 9.3. The dithering was started at $t = 1\,\text{s}$, and it was ramped up to the final set-point, both the center and the amplitude, to get a smooth transition. The ramping is further important, as the torque before the dithering is started may be close to the friction band boundary, as was the case in Fig. 9.5, and starting the dithering signal with the full amplitude, there would be a large risk of getting a torque outside of the friction band such that the robot would start to move. The middle diagram displays the control signals, i.e., the static torque feedforward and

the square wave amplitude. The bottom diagram shows the motor torque and the friction band estimate, and the center measurement signal is also displayed. The angular frequency of the dithering signal was chosen to be 12.7 Hz (80 rad/s).

The robot program was running with a sampling frequency of 250 Hz, but the dithering control loops were only updated every 10th sample, i.e., running at 25 Hz. The control signals were further kept constant when the measurement signals were close to the references, which here meant about 2 % of the estimated friction band. This means that the dithering signal can be seen as mostly a feedforward signal, and there will be time to detect applied external torques before the dithering control loops will start to eliminate the ”disturbance”, which the external torque will be interpreted as by the dithering control loops.

Comparison of using and not using dithering

Several experiments where external forces were applied to the robot in the Cartesian directions (see coordinate frame in Fig. 9.1) were performed to investigate which magnitude of forces that could be detected. Forces were applied in each direction approximately 50 times. The robot was positioned such that each applied force would result in torques influencing at least one joint, see Fig. 9.1. These experiments were first performed without using the dithering technique, i.e., the same type of experiment that was performed in Sec. 8.5, and some of the results are displayed in Fig. 9.6. Note that each subplot shows a separate experiment. In Fig. 9.7, a corresponding result from when dithering was used is displayed. All confidence interval estimates were calculated according to the method presented in Sec. 8.3, where $\lambda = 1.96$ such that a 95 % confidence interval would be given in the Gaussian case.

It can first be noted that when dithering was used, the force estimator was able to detect smaller forces in all directions. For the z-direction, however, the performance using dithering is only slightly improved as compared to without dithering. The second thing to notice is that the confidence interval estimate is tighter for the dithering case, and even a little bit too tight for the y-direction. Also the forces can in general be seen to be underestimated. The explanation to this is partly that the prior used for force estimation drives the estimate towards zero when the optimization problem (8.25) is solved; the measurement is explained as friction instead of external force. When dithering is used, part of the explanation also lies in the tendency of the method to underestimate the applied torque, as was displayed in Fig. 9.4. The underestimation effect gets reduced when the external forces become larger in relation to the friction torques, i.e., resulting in a decreased estimation error.

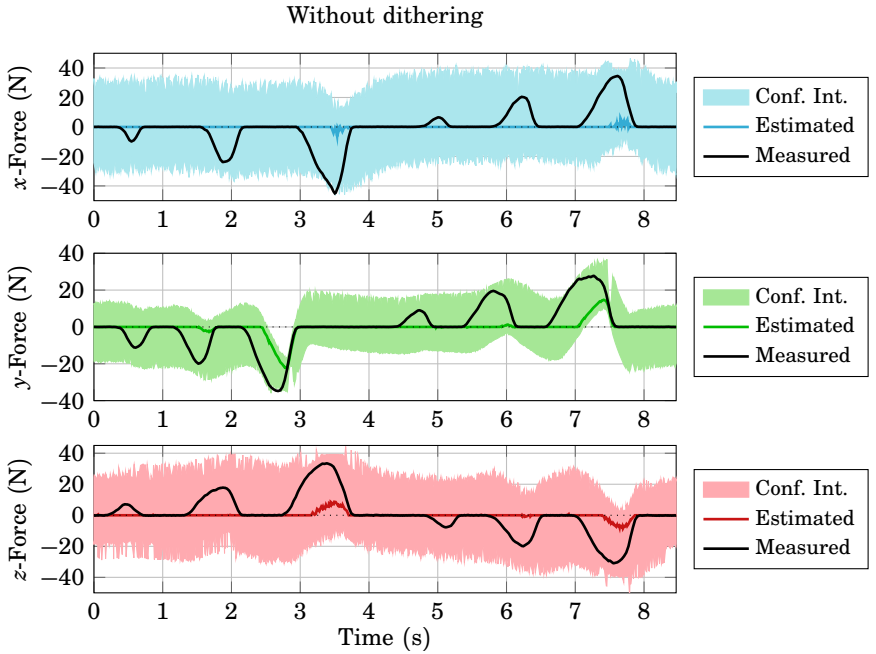


Figure 9.6 Estimated and measured forces from an experiment where external forces were applied to the end-effector of the robot, without using dithering.

The confidence interval estimate seems to be overly pessimistic for both cases, especially when the external force was zero. But when external forces were present, it can be seen that the confidence intervals seem to be more appropriate.

The confidence intervals sometimes seem to contain some information about the external force that is not visible in the actual force estimate, e.g., in the z-direction for the dithering case at $t = 2\,\text{s}$, $t = 5.7\,\text{s}$, and $t = 7\,\text{s}$. The reason for this not showing up in the force estimate is that it is absorbed by the remaining Coulomb friction, when solving the optimization problem (8.25). The Coulomb friction is modeled by a uniformly distributed random variable, as it may be anywhere within in the friction band. For the dithering case, however, the torque is controlled to stay in the center of the friction band. The uniformly distributed part of the disturbance torques can therefore be ignored in the optimization problem. Doing this removes uncertainty from the problem, which can be compensated for by increasing the variance of the noise parameter that is modeled as a Gaussian random variable. The increase of variance

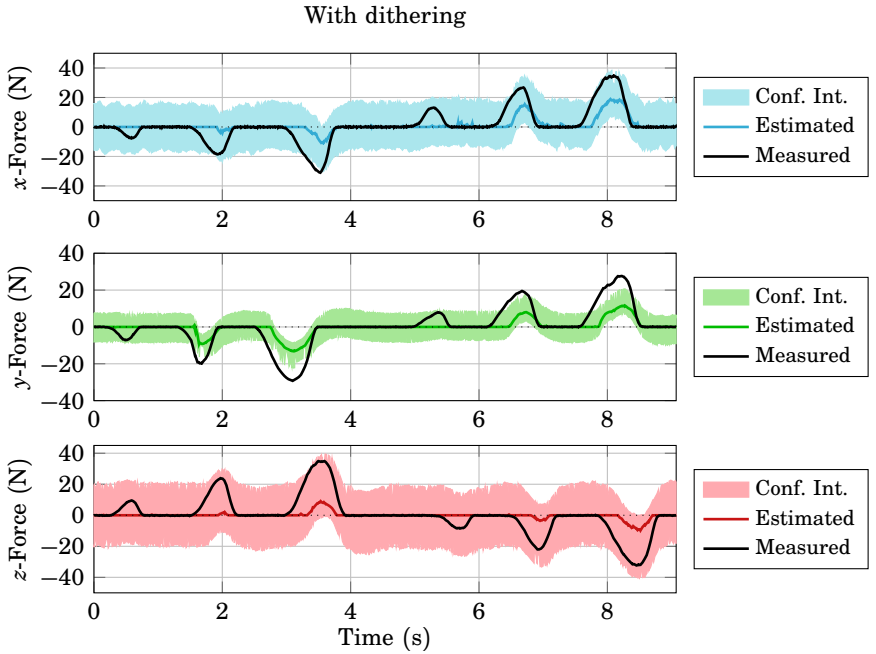


Figure 9.7 Estimated and measured forces from an experiment where the dithering method was used and external forces were applied to the end-effector of the robot.

can, for instance, be performed by approximating the Coulomb friction with a Gaussian random variable, uncorrelated with the noise term, and consider the new noise term as the sum of the old noise term and the approximated Coulomb friction. The result of doing this is displayed in Fig. 9.8. The estimation performance in the x- and y-directions are similar, but the z-force estimate has improved significantly and it is now as good as in the other directions. An effect of ignoring the uniform term was that non-zero force estimates when the external force was zero were much more frequent, e.g., the x-force at $t = 6\,\text{s}$ and the z-force at $t = 4\,\text{s}$.

Based on all performed experiments, 50 in each direction, both without and with dithering, it was investigated how large the applied force had to be to give a response that exceeded the noise level in the estimated force. Without dithering, no force smaller than 32 N gave any response in the x-direction and 22 N in the y- and z-directions. With dithering, responses could be seen in all directions already for forces with a magnitude of 10 N. It was also investigated how much force that was needed to estimate a force significantly different from zero, which here was chosen to be an

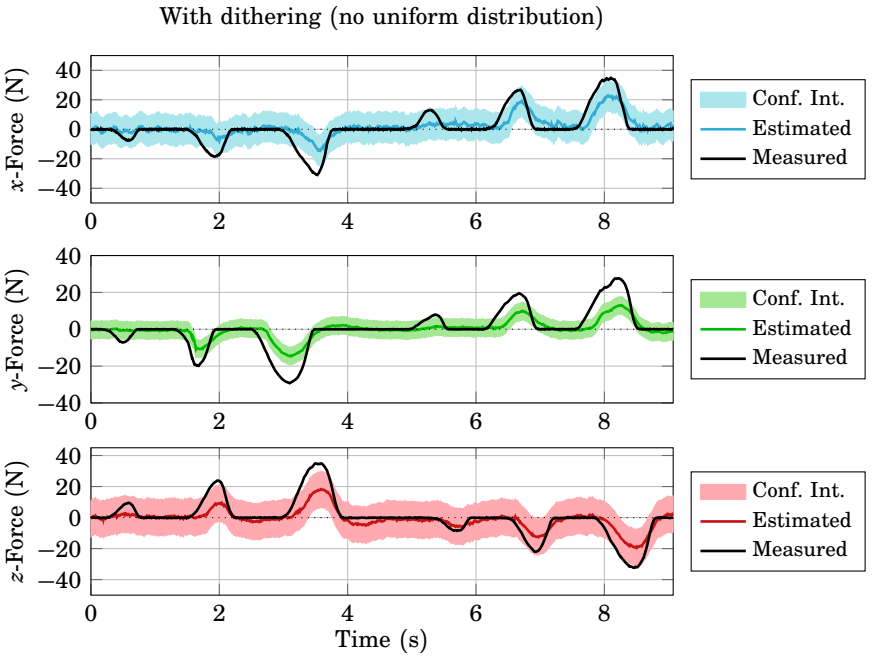


Figure 9.8 Estimated and measured forces from an experiment where external forces were applied to the end-effector of the robot. The dithering method was used without the Coulomb friction modeled as a uniformly distributed random variable.

estimated force above 10 N. Without dithering an applied force of 40 N was needed in the x-direction and 30 N in the y- and z-directions. When dithering was used, an applied force of 20 N was enough in all directions.

Lead-through programming

A lead-through program was implemented that was using the estimated external force. Dithering was initiated when the robot was not moving, and turned off as soon as a force was detected that resulted in that the robot started to move. The transitions between dithering and not dithering were made smooth by ramping the dithering signal up and down, as was displayed in Fig. 9.5.

Experimental data from a lead-through programming experiment are displayed in Fig. 9.9. The robot was first moved in the z-direction, then in the y-direction, and then in the x-direction before the experiment was ended with the robot being moved in several directions simultaneously. It can be seen that the estimated force in general follows the measured

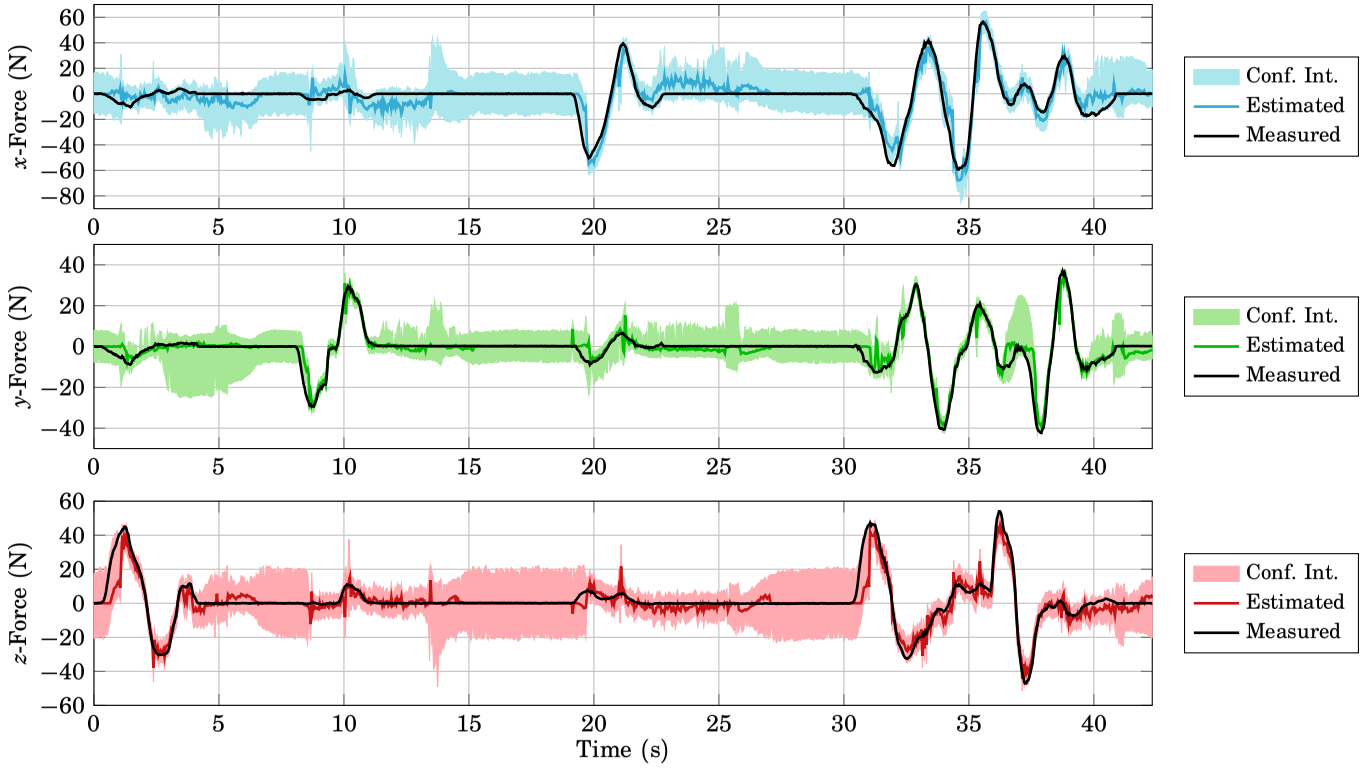


Figure 9.9 Estimated and measured forces (for verification) from a lead-through programming experiment.

force. During transfer from a zero-valued force to a non-zero value, it can be seen that the estimate is lagging behind somewhat, e.g., the z-force at $t = 1\,\text{s}$ and the x-force at $t = 20\,\text{s}$. This is an effect that can be seen in other experiments as well, for instance, in Fig. 9.4.

The confidence interval estimate becomes significantly tighter when the robot starts to move, see, for instance, between $t = 30\,\text{s}$ and $t = 40\,\text{s}$, where the width of the interval is approximately halved, compared to when the robot is not moving. The reason for the increased confidence is that the Coulomb friction is now assumed to be known, and only uncertainty in the Gaussian noise parameter remains. There are some false force estimates, e.g., the x-force estimate around $t = 12\,\text{s}$ and $t = 24\,\text{s}$. The confidence interval still contains the measured force, though.

9.6 Discussion

The dithering method presented in this chapter made it possible to detect external torques that were much smaller than the friction band. Without dithering, these external torques would most likely not have been noticed at all.

When using the dithering method, there is a risk that a too high amplitude of the torque feedforward signal is used. Doing that would result in a vibrating robot, which would result in anything but a user-friendly experience. The amplitudes chosen in this chapter were based on the measured motor angles, but also on the actual hands-on impression one got from the robot. Applying the dithering signal can be heard as a slight change in the sound coming from the motors, and it can be felt as a faint vibration when holding on to the end effector. It is, however, not discomforting. A person not aware of what is going on would probably not notice that much of difference from a robot not being applied to dithering. There is further a risk that the dithering signal will introduce unneccesary wear of the motors and the gear boxes. With the amplitude kept low and only used when needed, e.g., during lead-through programming, it should not be a problem.

The configuration of the robot is of high importance for the force estimation performance. If many joints can be used to estimate the force in a certain direction, then good estimates and tight confidence intervals can be expected, as information from many joints are combined. For the experiments in this chapter, the y-direction was a beneficial direction, while the other two, x and z, were slightly less beneficial. This could for instance be seen on the confidence interval estimates. When developing robot programs utilizing force estimation, the configuration of the robot is something the programmer has to bear in mind.

The dithering signal used in this chapter was a square wave. Other types of wave forms are also possible and it remains as future work to try them out. The frequency of the dithering signal is also something that should be further investigated. The frequency used was the same for all experiments, and chosen to provide a satisfactory torque response. When using a high-frequency signal there is a risk that aliasing becomes a problem, but no such problem has been observed.

The experiments performed in this chapter have shown that the torques due to external forces get underestimated. The reason for this could be that it is the motor torque reference that is the measurement, instead of the actual motor torque. For the reference to change, there have to be a position or velocity error, which may lead to a lower torque reference than the torque that was actually applied.

The applicability of force estimation depends on the task you intend to perform. Friction constitutes a fundamental difficulty that will give rise to estimation errors, which will limit the accuracy of force estimation. For the robot used in the experiments, an external force of around 10 N was needed for the estimator to notice it and around 20 N for it to be certain that there actually was a force when the robot was not moving. This is a significant improvement as compared to when no dithering was used, as forces with the magnitude of 20–30 N then were needed to get a reaction in the estimated force, and 30–40 N for the estimator to be certain that there actually was an external force present. Using dithering effectively decreases the Coulomb friction level, and increases the backdriveability of the robot. For another robot with more or less friction, the accuracy will be different.

Dithering was only used for joints that were not moving. The friction uncertainty for a moving joint was so small that dithering was unnecessary, and using dithering would probably only lead to a vibrating robot. The dithering method is thus only applicable in scenarios where the robot is not moving. The lead-through programming scenario investigated in this chapter is one such example. Another possible application is dual-arm operations where one of the arms only supports the second arm, e.g., a screwing task where one of the arms performs the screwing operation while the other only holds the pieces being screwed together.

The use of dithering was shown to increase the accuracy of force estimation in the case when the robot was not moving. The performance is, however, still quite far from that of a force sensor. In the lead-through programming scenario, the improved accuracy is hard to notice, as quite large forces usually are applied, so large that they would be detected without dithering as well. But for another scenario where it would be important to detect forces of small magnitude, much could be gained by using dithering.

Comparison to related work

A similar approach using dithering to reduce the effect of friction was presented in [H. Cho et al., 2014]. In that paper, a dual observer [Park and Lee, 2007] was used to estimate the external torque affecting each joint, where the joints were modeled as flexible two-mass systems and where the dynamics of the external torque was assumed to be known and given by a linear system. The Cartesian end effector force was then calculated by multiplying the estimated external torques with the inverse transposed robot Jacobian. This method needs identification of models for all joints, a model for the dynamics of the external torque, and also gravity compensation. Friction was not explicitly modeled.

To suppress friction in the joints, a sinusoidal dithering signal was sent as a torque feedforward signal. The interface to the robots used was similar to the one used in this thesis, at least according to available signals, but a difference is that the torque reference for each motor could be measured before the torque feedforward signal was added. The amplitude of the dithering signal was chosen to be equal to the estimated Coulomb friction level and the frequency as fast as the controller could handle, but how fast is not stated in the paper. The paper mentions nothing about vibrations, which were present for the robot used in this thesis if the dithering amplitude was chosen as large as the Coulomb friction level. The addition of the dithering signal should take away the effect of friction, and thus explaining why it has not been modeled. The paper mentions no difference in behavior for moving and static joints.

The experiments in the paper are not very well documented. All axis markers have been taken away, such that it is impossible to get any apprehension for the force estimation accuracy of the system. One experiment shows that dithering makes the system more responsive to applied external forces, such that the system reacts to forces with 20–36 % lower magnitude with dithering as compared to without dithering. These numbers can be compared to the performance of the method presented in this chapter, where a similar experiment showed that only about 50 % of the applied force was needed when dithering was used. Two different assembly experiments are described, but with no experimental data presented. The authors give very little guidelines for how all parameters of their method should be chosen, which makes it very difficult to make an implementation of their method and reproduce their results.

9.7 Conclusions

A method for improving the accuracy of force estimation when the robot is not moving was presented. It was based on using dithering as a feed-

forward torque signal. This made it possible to decrease the Coulomb friction uncertainty. Experimental results with an industrial robot verified that the method works, making it possible to detect forces of magnitudes around 10 N, as compared to 20–30 N when no dithering was used. An implementation in a lead-through programming scenario was also presented.

10

Sensorless Friction-Compensated Passive Lead-Through Programming

10.1 Introduction

Industrial robots have become indispensable in many places in industry today, such as the automotive industry. They have relieved human workers from hazardous and/or repetitive and monotone tasks, and they have increased the productivity and quality of the manufactured products due to their high speed and precision. The robots are usually placed in structured environments that are supposed to remain the same for a long time. This makes it worthwhile to put the required effort into performing the robot programming, which usually takes long time.

In other parts of the manufacturing industry, it is much more common with short-series production. For robots to be competitive here, the teaching phase must be quick and easy to perform to minimize the downtime. One easy way to accomplish a straightforward teaching method is to manually guide the robot, which is usually called lead-through programming or walk-through. This makes it possible for the robot to both learn positions and trajectories. It becomes especially convenient for the operator as no consideration of different coordinate frames etc. is needed. Lead-through programming for industrial robots is usually implemented by using force sensors. They are, however, often very expensive, and it would be preferable if lead-through programming could be accomplished in a sensorless setting.

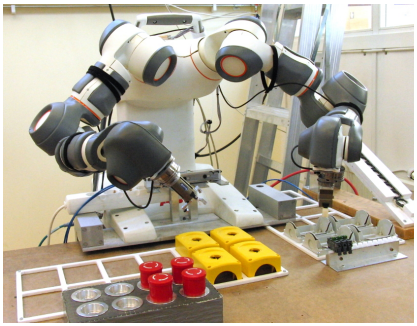



Figure 10.1 The robots used in experiments. Left: ABB YuMi. Right: ABB IRB120

A survey of different methods for programming industrial robots was presented in [Pan et al., 2012]. It was concluded that although online programming, i.e., using the joystick on the teach pendant to move the robot between positions to be used in the program, has several drawbacks, it is widely used. Further, several methods for incorporating sensors to simplify the online programming phase are described, e.g., using force sensors or vision systems. An application of lead-through programming is presented in [Ang et al., 2000], where lead-through programming was used to simplify the teaching of weld paths in a shipyard. To accomplish the lead-through programming, the robot was equipped with a force/torque sensor.

Programming by demonstration is a field where lead-through programming can be used for performing the demonstration, which sometimes also is called kinesthetic teaching. One such example is presented in [Wrede et al., 2013], where the authors focus on how to treat redundancy during kinesthetic teaching. The experiments were performed with a seven-degree-of-freedom robot, and a user study showed that it was beneficial to assist the operator by controlling the null-space of the robot, according to a redundancy resolution that was trained during an initialization phase. Another example of kinesthetic teaching is presented in [Calinon and Billard, 2007], where skills were taught to a robot in two steps. First, motion sensors were used to record the demonstrated task. Then the robot tried to perform the task and the teacher could interact through kinesthetic teaching, which was accomplished by choosing which motors of the robot that should become passive.

Methods for performing sensorless force control were reviewed in Chap. 8. An approach for detection of external torques without external sensing using the generalized momentum is [De Luca et al., 2006], where the method relies on knowledge of a dynamical model of the robot.

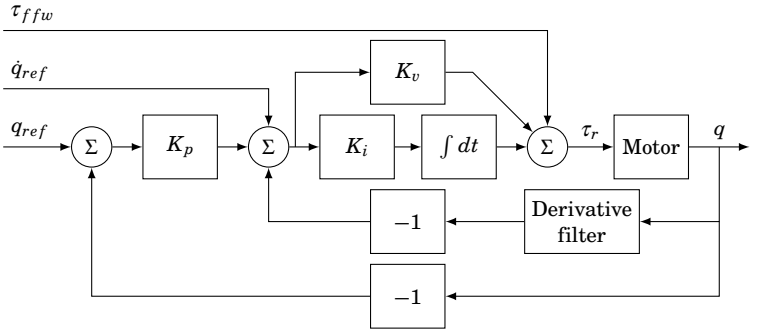


Figure 10.2 Schematic block diagram of the low-level joint controller running at 2 kHz.

This chapter will present a method for sensorless lead-through programming. It is based on disabling the low-level servo controllers in the joints and only feedforward the torques to balance gravity, i.e., the robot is in a passive mode with no position- or force-feedback loops running. This makes the interaction between the robot and any environment stable. It is further described how the lead-through programming performance can be improved by adding friction compensation. Experimental results from implementations on the ABB YuMi and ABB IRB120 are also presented, see Fig. 10.1 for the experimental setups.

10.2 Method

The robot system considered in this chapter has the control structure and interface described in Chap. 2. A schematic block diagram of a low-level joint controller is also given in Fig. 10.2.

Passive lead-through programming

Lead-through programming of the robot was accomplished by disabling the low-level joint control loops, i.e., setting the control gains K_p, K_v, and K_i to zero, see block diagram in Fig. 10.2. To prevent the robot from falling due to gravity forces, the torque feedforward signal was used to apply the motor torques needed to counteract gravity. As the joint torques commanded were purely based on feedforward, external forces applied to the robot could lead to movements if they exceeded the friction forces in the joints. The friction forces, mainly Coulomb and viscous friction, were helpful, as they made sure that the robot did not move when no external forces were applied. The lead-through programming worked in the same way as releasing the brakes of the robot, while maintaining feedforward

gravity compensation. As no feedback loops were active, interaction with the environment became stable without the need for tuning any control parameters.

Note that the lead-through programming was implemented independently for each joint, i.e., the implementation was made in joint space.

Gravity compensation

The rigid body dynamic model of a robot is linear in the parameters [Siciliano et al., 2009]. The values of these parameters for the robots used in experiments were, however, not known a priori and therefore needed to be identified. Identifying all of the dynamic parameters is difficult, as the experiments to be performed must be chosen with care to be sufficiently exciting to make all parameters identifiable. For the intended application, namely lead-through programming, the robot is expected to move relatively slowly with low accelerations, which means that the dynamic torques will be small, and they were therefore neglected by setting all velocities and accelerations to zero when deriving the equations. The remainder of the dynamics model was therefore only related to gravity, with four parameters representing the mass and a vector to the center of gravity for each link of the robot. A simple friction model was also added, where the friction torque, τ_{fric}, was modeled as

$$\tau_{fric} = \theta_C \text{sign}(\dot{q}) + \theta_v \dot{q} \tag{10.1}$$

where $\dot{q}$ is the joint velocity, θ_C is the Coulomb friction, and θ_v the viscous friction parameter. Each link had thus in total six parameters, and as the model was linear in the parameters, they could be estimated using the least-squares procedure with data from an experiment where the robot slowly moved around in its workspace.

Torque feedforward control

The gravity compensation torque was sent as a torque feedforward signal to each of the low-level joint control loops. Due to the fact that the control gain K_i is situated before the integral in Fig. 10.2, setting it to zero will just stop the update of the integral state, but it will still hold its value. This means that the feedforward torque signal will have to compensate for this offset. To be certain that the desired torque was actuated, a combined feedforward and feedback strategy was applied, see a block diagram of the control loop in Fig. 10.3. The reference signal, denoted by τ_{ref}, is the desired torque, i.e., the torque due to gravity, the measurement signal is the torque reference sent to the motor, τ_r, and the control signal is the feedforward signal, τ_{ffw}.

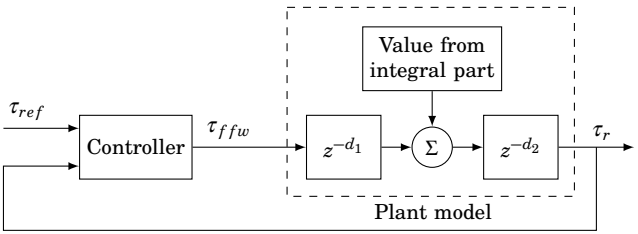


Figure 10.3 Block diagram of the torque feedforward control loop, running at 250 Hz. In the plant model, z denotes the shift operator. The delays d_1 and d_2 are unknown, but it is known that $d_1 + d_2 = 3$.

In the research interface used, there were delays both when references were set and when measurements were received. These individual delays were unknown, whereas their sum was estimated to be three sampling periods. A model with these properties is given as the plant model in Fig. 10.3. To account for the delays, the following modified measurement signal was used

$$y_{mod}(t) = \tau_r(t) + \sum_{k=1}^{2} \Delta u(t - kh) \tag{10.2}$$

where $\Delta u(t)$ denotes the the update of the control signal at time t, and h the sampling period. Using $y_{mod}(t)$ for feedback is the same as using a Smith predictor [Smith, 1957], where the process model is a time delay. An integral controller was used to close the loop, i.e., the torque feedforward signal was calculated as

$$\tau_{ffw}(t) = \tau_{ffw}(t - h) + \Delta u(t) \tag{10.3}$$

where $\Delta u(t)$, the update of the control signal, was

$$\Delta u(t) = \underbrace{(\tau_{ref}(t) - \tau_{ref}(t - h))}_{\text{feedforward}} + \underbrace{K\,(\tau_{ref}(t) - y_{mod}(t))}_{\text{feedback}} \tag{10.4}$$

where K is the feedback gain of the controller, and $\tau_{ref} = \tau_G$, i.e., the reference is the torque due to gravity, τ_G. The feedforward part handles most of the reference changes, and the feedback part handles disturbances, i.e., the value held in the integral part as shown in Fig. 10.3. Note that this control loop is active only when the low-level control loop (Fig. 10.2) is inactive, and vice versa.

Friction compensated passive lead-through programming

Friction torques in the joints are helpful as they prevent the robot from moving except for when external torques are applied. At the same time,

the friction torques make it heavy to move the robot, especially if the friction torques are large. To make it easier to move the robot, additional torque feedforward can be used to compensate for friction based on movement of the robot. Ideally, it should be possible to use the torque due to gravity plus the Coulomb friction torque. Then the robot would be truly free-floating. In reality, neither the torques due to gravity nor the Coulomb friction will be exactly known, and to feedforward the entire Coulomb friction estimate might result in an accelerating joint without any external torque applied. There should still exist some friction that stops the robot when no external forces are present and the residual friction required depends on the viscous friction level and the quality of the Coulomb friction estimate.

The velocity measurement in the robot is based on numerical differentiation of the position measurement, and consequently the standard deviation of the noise of the speed signal will be a factor $\sqrt{2}/h$ larger than the position measurement noise, where the sampling period $h = 0.004$ s in our case. The obtained velocity is further low-pass filtered, which somewhat decreases the noise level. The friction compensation is added once the velocity exceeds a threshold, which must be chosen with respect to the noise level. To make a smooth transition when the torque feedforward is added, the amount of friction compensation is made proportional to the velocity for small velocities. Specifically, the friction compensation torque, τ_{FC}, was calculated as

$$\tau_{FC} = \begin{cases} 0 & , \quad |\dot{q}| < \dot{q}_0 \\ \frac{|\dot{q}|-\dot{q}_0}{\dot{q}_1-\dot{q}_0}\text{sign}(\dot{q})a\hat{\theta}_C & , \quad \dot{q}_0 \leq |\dot{q}| < \dot{q}_1 \\ \text{sign}(\dot{q})a\hat{\theta}_C & , \quad |\dot{q}| \geq \dot{q}_1 \end{cases} \tag{10.5}$$

where $\dot{q}_0$ and $\dot{q}_1$ are the velocity thresholds defining the proportional region, and a is the percentage of the Coulomb friction estimate $\hat{\theta}_C$ that is used as friction compensation. The velocity may increase fast, and the friction compensation will then almost be in the form of a step, which could be unpleasant for the operator since it would feel like an abrupt change in the resistance of the joint in question. This situation was avoided by also limiting the rate of change of τ_{FC}.

The feedback control loop for the torque feedforward signal generation is useful now, as the friction compensation torque can be handled by just adding it to the reference, i.e., use $\tau_{ref} = \tau_G + \tau_{FC}$ in Eqs. (10.3)–(10.4).

Increased sensitivity to external torques when a joint is not moving

A major difficulty with sensorless lead-through programming is that the Coulomb friction (or stiction) must be overcome to start moving the robot.

This effect can be experienced both when the lead-through programming is started from rest, and when one wants to move the end effector of the robot while manually keeping the orientation fixed. In the latter scenario, many joints will have to move simultaneously and there will most likely be joints where the lever arm from the end effector is short, which means that quite some force will be needed to overcome the friction in those joints.

It was experimentally noted that it was possible to achieve an increased sensitivity to external torques by activating the low-level joint controllers with an increased integral gain when the joints were not moving [Lundberg, 2013]. In the experiments performed in this thesis, 100 times the nominal value of the integral gain was used, and then it became possible to detect external torques that were significantly smaller than the Coulomb friction level, see further the experimental results section.

The idea is to activate the controller with high integral gain only when a joint is not moving, and when an external torque is detected it should be turned off again. If the detected torque is small, it will not overcome the friction torque. Therefore, a short torque feedforward pulse is commanded, and if the operator really intended to move the joint, i.e., keeps applying a force, the pulse will help the joint start moving. Once the joint is moving, the friction compensation, according to (10.5), will be active and help the operator.

The detection of external torques is performed by using detection thresholds. The torque measurement, τ_r in Fig. 10.2, might end up in other places than in the middle of the friction band when the controller with high integral gain is activated. To account for this, the thresholds were centered around a delayed and filtered version of τ_r. To handle the initialization phase, when there was uncertainty of where τ_r would end up inside the friction band, the thresholds were ramped down from the Coulomb friction estimates to the final thresholds. The upper threshold, Λ_{up}, was calculated as

$$\Lambda_{up}(t) = \begin{cases} \hat{\tau}_{center} + \dfrac{t\lambda + (T-t)(\hat{\tau}_G + \hat{\theta}_C)}{T} & , \quad t < T \\ \hat{\tau}_{center} + \lambda & , \quad t \geq T \end{cases} \tag{10.6}$$

where the time t is assumed to be zero when the initialization starts, T is the length of the initialization phase, $\hat{\tau}_{center}$ is a low-pass filtered and delayed version of τ_r, λ is the final threshold, $\hat{\tau}_G$ the estimated gravity torque, and $\hat{\theta}_C$ the estimated Coulomb friction. The lower threshold was calculated analogously.

Small movements

Lead-through programming based on the features described earlier in this section works very well for large movements. However, if the operator is interested in doing small adjustments of the end effector, e.g., learning a position for gripping an object, the friction compensation scheme becomes counterproductive. An attempt to move the end effector a small distance might result in a larger movement when the friction compensation torques are activated. The reason is that at first, the operator applies a force to start the movement, and as this force is sensed, extra help in the form of friction compensation is activated, and then the applied force is too large and the movement becomes larger than intended.

For small movements, it was found out in experiments that it is in practice better to disable all friction compensation. This will make it harder to move the robot, as the operator will have to overcome all friction forces. On the other hand, the constant friction resistance makes the friction predictable and manageable.

The lead-through programming implementation will have to switch between friction compensation on and off to work well for both large and small movements. One way to do this is to investigate the maximum distance moved by the end effector during a fixed time window. Small movements will almost always be fine adjustments of the end effector, e.g., fine tuning a gripping position, and the end effector movement is therefore the relevant measure to use. To be precise, the measure used for determining when to switch between friction compensation on and off is defined as

$$\text{maxDist} = \max_{t_0 \in [t-\Delta t;t]} \|p(t) - p(t_0)\|_2 \tag{10.7}$$

where $p(t)$ denotes the Cartesian position of the end effector at time t, and Δt is the time window used. To make the transition smooth, a linear region where the amount of friction compensation is proportional to the value of maxDist was also introduced, i.e., using the friction compensation torque $\tau_{FC}^{new} = b\tau_{FC}$, where τ_{FC} is defined in (10.5) and the factor b as

$$b = \begin{cases} 0 & , \quad \text{maxDist} < d_1 \\ \dfrac{d_2 - \text{maxDist}}{d_2 - d_1} & , \quad d_1 \leq \text{maxDist} < d_2 \\ 1 & , \quad \text{maxDist} \geq d_2 \end{cases} \tag{10.8}$$

where d_1 and d_2 defines the linear region.

10.3 Implementation

Two robots were used for experiments, see photos in Fig. 10.1. The first one was the ABB YuMi and the other robot was the ABB IRB120. The

YuMi robot was equipped with wrist-mounted ATI Mini 40, 6 degree-of-freedom force/torque sensors, which made it possible to collect validation data.

Demonstrators

The lead-through programming was implemented in two different versions. The first one was such that the operator teaches a number of positions, and when the taught program is being replayed, the trajectory between the positions is planned by the native robot controller. For small movements (the end effector moving less than a certain distance), the robot first tries to move its end effector linearly, and if that is not possible the fallback is a joint move, i.e., performing a linear motion in joint space instead of in Cartesian space.

The other demonstrator records the actual lead-through programming trajectory, by saving positions with a frequency of 10 Hz. When the learned program is being replayed, the native controller performs joint moves between the recorded positions, with possibilities both to increase and to decrease the velocity of the motion.

10.4 Experimental results

The experiments described in this section were made with the ABB YuMi, unless otherwise stated.

Parameters

The parameters for the gravity compensation and for the friction model were estimated from an experiment where the robot slowly moved around in its workspace without any interaction with the environment. As the friction model (10.1) is only valid for velocities different from zero, data from the experiment where the velocity was close to zero were excluded when performing the parameter identification. The resulting compensation gave a mean absolute error ranging from 0.3 Nm for the base joints to 0.03 Nm for the wrist joints.

The friction compensation parameters in (10.5) were tuned manually. The lower velocity level, $\dot{q}_0$, was chosen such that the noise in the velocity measurement did not trigger any torque feedforward, and the upper level, $\dot{q}_1$, was chosen such that the transition from no torque feedforward to full torque feedforward felt smooth when manually guiding the robot. The parameter a, the percentage of the estimated Coulomb friction to use as feedforward, was chosen as high as possible without getting any drifting joints. This magnitude depended on the accuracy of the gravity

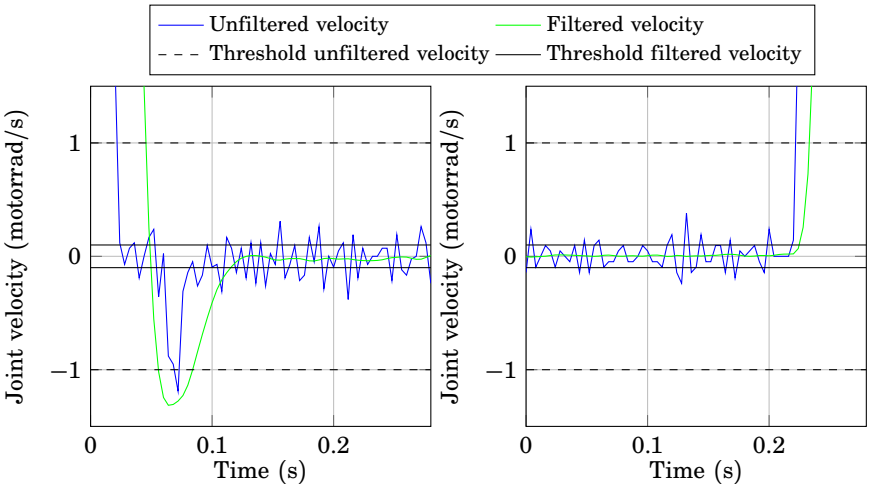


Figure 10.4 Illustration of the threshold levels used for activating the joint controllers with large integral gain. Both the filtered and the unfiltered velocity must be below their respective threshold for the large integral gain to be activated. When the large integral gain is active, it is deactivated if any of the thresholds is exceeded. The left diagram displays an example of when a joint stops moving, and the right diagram an example of when a joint starts moving.

compensation and the estimated Coulomb friction, and for the YuMi robot 60–80 % of the estimated Coulomb friction was used.

The parameters for switching the friction compensation on and off used in (10.7)–(10.8) were chosen as $\Delta t = 0.5\,\text{s}$, and the start of the proportional region as $d_1 = 2\,\text{cm}$, and full friction compensation was active at $d_2 = 5\,\text{cm}$.

The other parameters used are described in the following subsections.

The use of large integral gain

The integral gain, K_i, was increased when the velocity became low. To get rid of the noise in the velocity measurement, it was low-pass filtered, such that a low threshold could be used. Filtering, however, delays detection of the start of a joint movement, and therefore also the unfiltered velocity measurement was thresholded, but with a significantly higher threshold. Illustrations of the thresholds are displayed in Fig. 10.4. To increase K_i, the velocity had to be below both of the above mentioned thresholds. Further, to get some robustness towards when the velocity changes sign, the velocity had to be below the thresholds for 0.1 seconds for the controller to

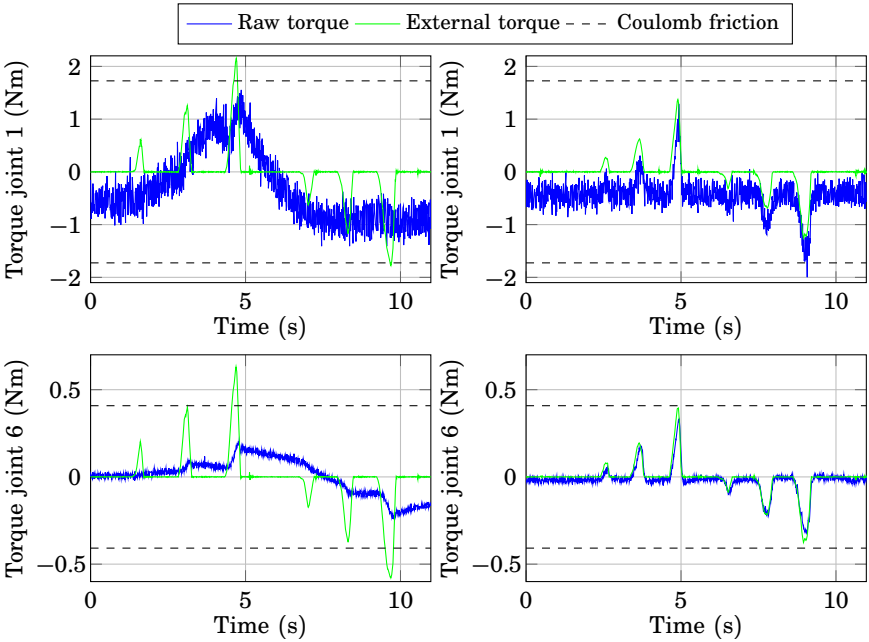


Figure 10.5 Experiment where forces were applied to the end effector of YuMi. Results from two joints are shown; the top row shows the result for the first joint, and the bottom row shows the results for the sixth joint[1]. The left subplots show the raw torque response for the nominal controller parameters and the right subplots the response for the case with high integral gain. The external torque has been measured with a wrist-mounted force/torque sensor.

be activated. The controller with large integral gain resulted in a stable system as long as the joint remained at rest. At rare occasions, though, the noise triggered a motion that led to instability of the system. Therefore, the velocity was supervised in this phase, and the large integral gain was turned off if the velocity exceeded any of the thresholds previously mentioned. When K_i was increased, the other controller parameters, K_p and K_v, had their nominal values.

The benefit of using the controller with a large integral gain is displayed in Fig. 10.5. It shows an experiment where forces were applied to the end effector of the robot. The experiment was carried out twice; the left subplots show the case with nominal controller values and the right subplots the case where K_i is 100 times larger than its nominal value. The

[1] With the ABB convention, the sixth joint is called joint five.

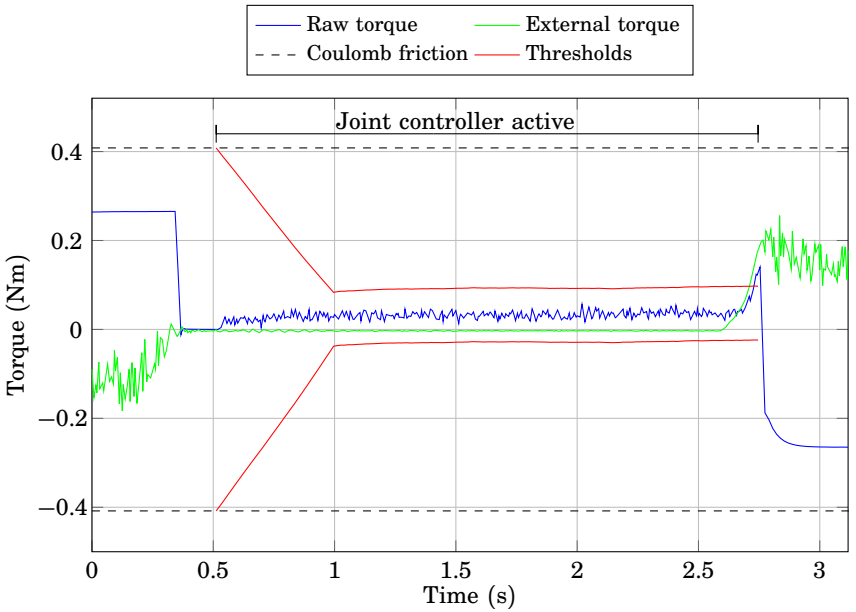


Figure 10.6 An illustration of the behavior of the detection thresholds used when the joint controllers with high integral gain were active. At $t = 2.7$ s an external torque was applied to the robot, which the motor initially counteracted, as can be seen in raw torque curve. When the threshold was exceeded, a friction compensation torque was applied. The experiment was performed with the sixth joint of YuMi[2].

upper plots show the response in the first joint of the robot, and the lower plots the response for the sixth joint. A wrist-mounted force/torque sensor was used to give the validation measurement. The torque due to gravity has been compensated for in the plots. It can clearly be seen that using the large integral gain is beneficial, as external torques within the estimated Coulomb friction band are clearly visible in the raw torque data, i.e., the signal denoted τ_r in Fig. 10.2. Without the use of high integral gain (the left subplots), the raw torque is unpredictable with drifting curves, and it would be very difficult to detect the applied forces without the validation data. It can further be noted that the noise level is much lower for the sixth joint, and also the other wrist joints, making it possible to detect lower torques for these joints. The reason for this is probably that the base joints support a larger part of the robot structure, with more noise due to mechanical resonances.

[2] With the ABB convention, the sixth joint is called joint five.

An experiment that illustrates the behavior of the detection thresholds used for detecting external torques is displayed in Fig. 10.6, where the thresholds are displayed in red and the Coulomb friction estimate in the dashed black lines. In the beginning of the experiment, the joint was moving and a torque feedforward of 65 % of the friction band was applied, which can be seen in the raw torque curve (blue). The applied torque can also be seen in the external torque signal (validation data from a force sensor). When the external torque disappeared (at $t = 0.4\,\text{s}$), the joint stopped moving and the torque feedforward was stopped as well. After the velocity had been below the thresholds illustrated in Fig. 10.4, the joint controller with the high integral gain was activated, as indicated in the top of the plot. The raw torque shows a slight positive drift, which was captured by the thresholds. At $t = 2.7\,\text{s}$, an external torque appeared. The motor first counteracted the external torque, i.e., tried to keep the position of the joint, but when the detection was made, a helping torque to compensate for friction was sent as feedforward. Initially, the feedforward was in the form of a pulse with a duration of 0.2 s, as described in Sec. 10.2. The length of the pulse was chosen such that the robot had time to start moving in case the operator would keep applying a force, i.e., such that the velocity-based friction compensation (10.5) was activated. In Fig. 10.6, this is exactly what happens. The amplitude of the pulse was chosen to be the same as the velocity based friction compensation, which explains why the pulse can not be separated from the velocity-based friction compensation. When the detection of the external torque was made, the joint controller was deactivated.

The parameters used for the detection thresholds (10.6) were chosen such that the initialization phase was $T = 0.5\,\text{s}$, and the final threshold λ somewhat larger than the noise level as can be seen in Fig. 10.6 between $t = 1\,\text{s}$ and $t = 2.7\,\text{s}$. The center of the threshold levels was taken as the 40 samples (0.16 s) delayed mean of 80 samples (0.32 s) of the raw torque signal.

Torque feedforward control

An example of the performance of the torque feedforward controlller (described on page 189) is displayed in Fig. 10.7, which displays the behavior of the third joint of YuMi when an external torque was detected and a friction compensation torque was commanded. In the beginning of the experiment, for times less than t^*, the joint controller with high integral gain was active, i.e., the torque feedforward controller was inactive. At time t^*, an external force was detected and a torque feedforward of 80 % of the estimated friction band was commanded, as can be seen in the reference signal τ_{ref}. As was described in Sec. 10.2, the feedforward part

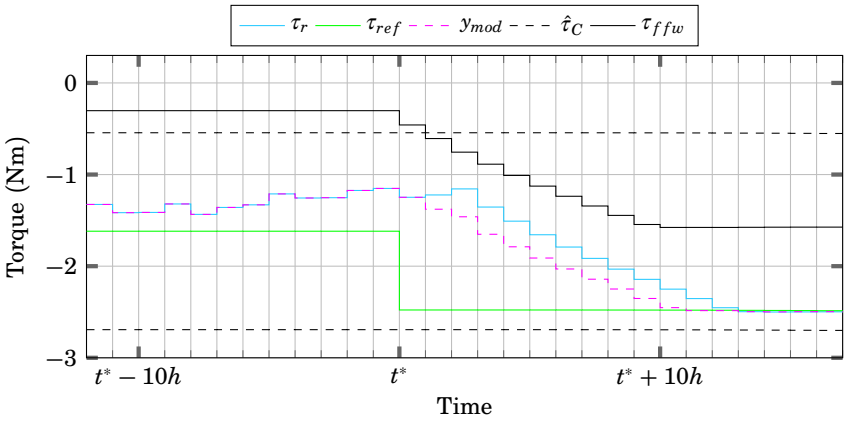


Figure 10.7 Experiment that illustrates the behavior of the feedforward control loop for one joint (the third joint of YuMi[3]). An external torque that generated a feedforward torque was detected at time t^*. The sample period $h = 4$ ms.

of the controller handles most of the set-point change, as can be seen in the fast response in the control signal, τ_{ffw}. The limited rate-of-change of τ_{ffw} can also clearly be seen, as τ_{ffw} was ramped down instead of being changed in a step. The velocity increased quickly in this experiment, and the region where the friction compensation torque was proportional to the velocity as defined in (10.5) is not visible. By comparing τ_r and τ_{ffw}, the delay can be noticed, and it can further be seen that y_{mod} (defined in Eq. (10.2)) works as an approximation of τ_r without the delay.

The feedback component of the controller takes care of the deviation of τ_r from the reference, while the feedforward part handles the reference change. The feedback gain, K in (10.4), was manually tuned such that the controller was fast but without getting any overshoot when the controller was activated, e.g., which happened at time t^* in Fig. 10.7.

Lead-through programming performance

Experiments were carried out to investigate the lead-through programming performance. The first experiment was performed to show how much external torque was required to start moving a joint by manually applying an increasing force until the investigated joint started to move. The experiment was carried out 30 times without friction compensation, and another 30 times with friction compensation. Two different joints were in-

[3] With the ABB convention, the third joint is called joint seven.

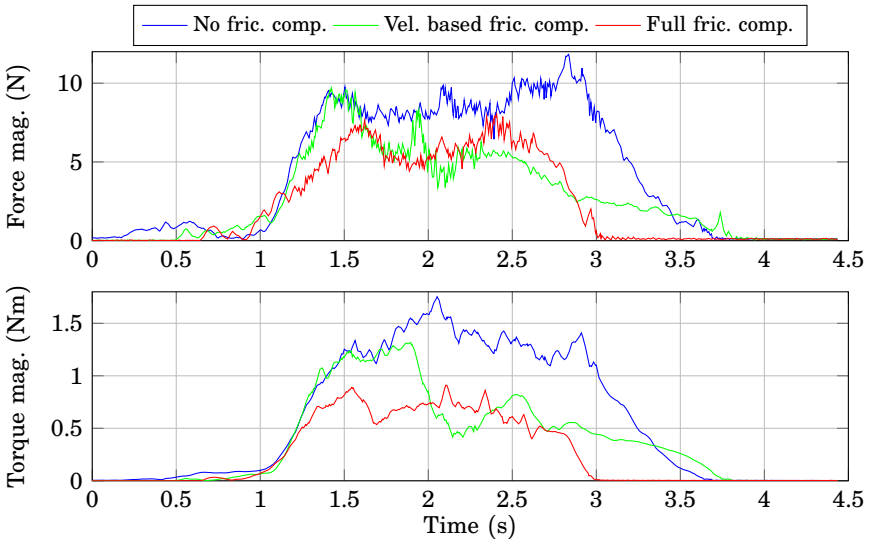


Figure 10.8 Results from an experiment where the end effector of YuMi was moved linearly while trying to keep the orientation fixed. The top diagram shows the measured force magnitude from the wrist-mounted force/torque sensor, and the bottom plot shows the torque magnitude. The experiment was performed three times, first with no friction compensation, then with friction compensation based on velocity only according to (10.5), and finally with the full friction compensation.

vestigated, the first joint to represent the base joints, and the sixth joint[4] to represent the wrist joints. For the first joint, on average only 55 % of the applied torque was needed when friction compensation was active as compared to when it was not. The standard deviation was 9 percentage points. For the sixth joint, the corresponding result was that only 40 % of the torque was needed, with a standard deviation of 15 percentage points. The main reason for the difference in benefit was the larger noise level for the base joints. When moving the end effector, however, the lever arm is longer for the base joints, and this compensates for the fact that larger torques are needed.

Another experiment analyzed friction compensation for linear end-effector motion. The experiment was performed by manually moving the end effector while trying to keep the orientation fixed. Results from this experiment are displayed in Fig. 10.8, which shows the force and torque

[4] With the ABB convention, the sixth joint is called joint five.

magnitude from the wrist-mounted force/torque sensor. The experiment was performed three times; first with no friction compensation, then with friction compensation based on the measured velocity, i.e., according to (10.5), and finally with the full friction compensation, where also the method with large integral gain was used. Without friction compensation, the largest applied force and torque was needed, as can be seen in the diagrams. Using friction compensation based on measured velocity initially required the same force and torque to start the robot movement, but after the initial transient when the friction compensation torques were applied, lower external forces and torques were needed. For the full friction compensation, moving the robot was easier, which can be seen in Fig. 10.8 as lower applied force and torque. Especially, the lower applied torque felt convenient when moving the robot.

The lead-through programming was compared to using the joystick on the teach pendant to teach a simple task, namely to pick an object. Three positions needed to be taught; a position above the object to pick, a position where the gripper can close around the object, and a position where the robot can safely move away to with the object. Using lead-through programming, teaching these three positions took 25 s, but using the teach pendant took over 2 minutes. The difficult part using the teach pendant was to move the robot to the correct orientation to be able to pick the object. Thus, lead-through programming can substantially decrease the amount of time needed for teaching a robot program.

Lead-through programming of IRB120

The lead-through programming was also implemented on the ABB IRB120, which has significantly more friction than YuMi. The gravity compensation resulted in a mean absolute error that was 1 Nm for the base joints and 0.1 Nm for the wrist joints. In contrast to YuMi, the IRB120 has a significant amount of viscous friction in the joints, and this made it possible to feedforward a larger amount of the Coulomb friction torque, and 80–100 % of the estimated Coulomb friction was used for feedforward. Unfortunately, the IRB120 was not equipped with a force sensor and, therefore, no validation data are available. An experiment was performed to show that the benefit of using high integral gain was valid also for this robot, see Fig. 10.9. Forces were manually applied to the end effector; the left subplots show the result for the nominal controller gains and the right subplots for when high integral gain was used. The qualitative behavior is the same as the experiment performed with YuMi presented in Fig. 10.5, but without validation it is difficult to say more.

The lead-through programming was implemented in the same way as for YuMi, and it works well joint by joint. Moving the robot in Cartesian

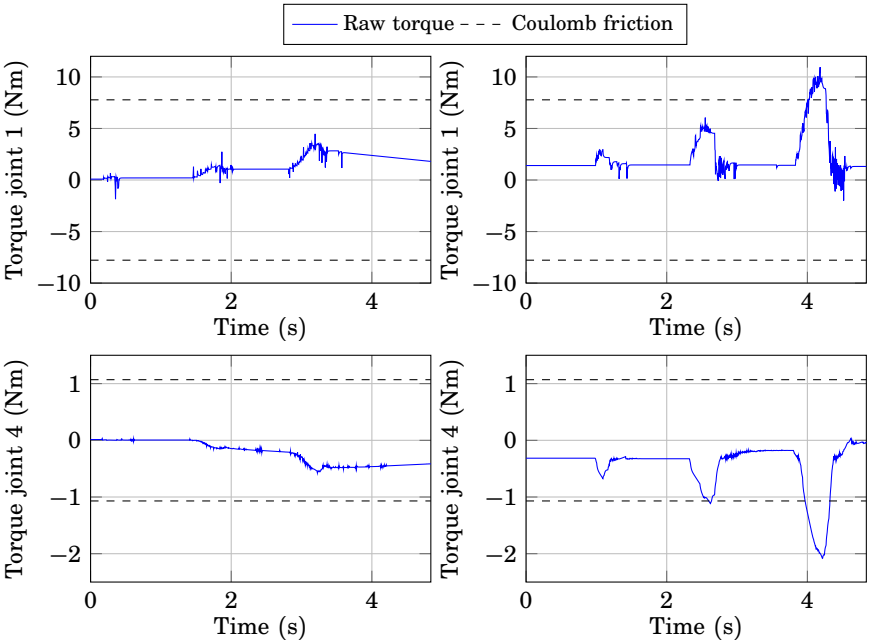


Figure 10.9 Experiment where forces were applied to the end effector of IRB120. The results are shown for two of the joints; for the first joint in the top row, and for the fourth joint in the bottom row. The left subplots show the raw torque response for the nominal controller parameters and the right subplots the response for the case with high integral gain. No validation data are available as no force sensor for validation was mounted on the robot. The manually applied forces were intended to be equally large in both the left and the right subplots.

directions becomes quite hard, as a lot of force is required to move all the joints that need to move, also without the strategy for turning off the friction compensation for small movements defined in (10.7)–(10.8). The main reason for this was the viscous friction. As compared to the YuMi implementation, the lead-through programming with IRB120 was not as good, but it would still be useful for teaching robot programs.

10.5 Discussion

The lead-through programming described in this chapter was passive, in the sense that no force-feedback control loops were used. An alternative implementation would be to make it active, i.e., such that each joint is actively controlled. In the active approach, the problem is about estimat-

ing the external forces in the presence of the friction disturbances, rather than compensating for friction. A drawback with that method is that it is difficult to get good performance in both free-space motions and when the robot is in contact with the environment. A controller that performs well in free-space motion may become unstable in contacts with stiff environments. Forces may build up quickly in contact with a stiff environment, and the dynamics of the robot and time delays that exist in the control system, in spite of being small, become problematic for the feedback interconnection [An and Hollerbach, 1987]. Stable controllers for stiff environments can be designed, e.g., using the notion of passivity, and the performance can be increased by also modeling the environment [Buerger and Hogan, 2006]. On the other hand, a controller performing well during stiff contacts can be sluggish and hard to move during free-space motions. A controller that switches between two different parameter settings can solve this problem, but it would be difficult to automate the switching and making the switching manually would decrease the user-friendliness. The passive lead-through programming contains no feedback loops and hence does not suffer from this drawback.

The presented lead-through programming implementation contained a lot of different switches, i.e., it is based on a hybrid control approach [Lunze and Lamnabhi-Lagarrigue, 2009]. A number of parameters have to be tuned, which may make it difficult to use the method. On the other hand, as each joint can be tuned individually, it is quite easy to perform the tuning procedure. The implementation in this chapter was based on the ABB research interface, but the implementation should be possible to do with other robots and interfaces with similar performance, such as the KUKA-FRI and Comau C5G Open.

Increasing the integral gain was only possible when a joint was not moving. The larger gain resulted in instability when the joint was moving. By using speed supervision, i.e., only using the large integral gain when the velocity was zero, the resulting system became stable. Whereas the reason why the increased sensitivity of external torques is possible is not fully determined, a hypothesis is that the effect is coupled with the fact that the investigated robots have harmonic drive gears, as the effect has not been observed for robots with other types of gears. The large integral gain increases the bandwidth of the control loop. The hypothesis is that the increased bandwidth together with the measurement noise functions as a dithering signal, which would mean that the motor is constantly moving and thereby has a significantly lower resulting friction torque. The effect of the large integral gain had a similar effect as the dithering method described in Chap. 9. The large integral gain could, however, not be used to replace the dithering method for IRB140, which have a different type of gear box and therefore did not exhibit the same effect as YuMi and

IRB120.

The lead-through programming was implemented in joint space, i.e., each joint of the robot moved independently. One benefit of doing this is that no problems with singularities of the robot will occur. A disadvantage is that in a purely passive approach it is not possible to make the robot keep the end-effector orientation and only move linearly. On the other hand, with the friction compensation, it was fairly easy to manually fix the orientation of the end effector while moving the robot. For a redundant robot like YuMi, lead-through programming implemented in task space would further have to control the redundant degrees of freedom. With the presented joint-space lead-through programming, it was quite intuitive to use two hands; one to take the end effector to the desired position, and the other to move the elbow of the robot accordingly.

Using a force sensor for implementing lead-through programming has the advantage that problems with friction can be avoided, provided that the friction in the joints is handled by the low-level joint controllers. The sensor is usually attached to the wrist of the robot and it can be used to perform lead-through programming of the end effector. Any redundant degrees of freedom will have to be taken care of by the controller, and only forces applied that the force sensor can measure will give rise to motions of the robot. With a wrist-mounted sensor, forces applied to the robot arm inside of the wrist can not be measured and will hence not lead to any motion of the robot. As the implementation of lead-through programming with a force sensor most commonly is the active version, the problem with instability in contact with stiff environments is also present.

A simple friction model, Eq. (10.1), was used in this work. For the YuMi robot, however, there are quite large position-dependent friction torque variations [Bagge Carlson et al., 2015]. This is the main reason why only a relatively small percentage of the Coulomb friction torque could be used as friction compensation. By using a position-dependent friction model for each joint it should thus be possible to use a larger feedforward torque, and hence improve the lead-through programming performance. This remains as future work.

The implemented lead-through programming was based on recording positions and then to use the functionality in the native robot controller to move between the saved positions. For the mode where a continuous trajectory was recorded, future work includes putting more effort into making the trajectories smooth. As of now, the trajectories sometimes become a little bit jagged, as the trajectory generator only considers the two positions the robot is moving in between when planning the path.

The proposed lead-through programming was evaluated experimentally on two different robots. Lead-through programming with YuMi worked better, mainly due to significantly lower amount of friction present

in the joints. For lead-through programming purposes in particular, the benefit of a force sensor increases with the amount of friction in the robot, which is usually correlated with the size of the robot. Overcoming the friction with a sensorless approach for a large robot may demand too large forces, but with a force sensor, the operator only has to apply a force that is larger than the noise level of the sensor.

10.6 Conclusions

A method for performing sensorless lead-through programming with industrial robots was presented. The method works by disabling the low-level joint controllers and only feedforward the torque due to gravity. It was reported how friction compensation could be added based on the measured velocity, which was shown to decrease the external force needed from the operator to move the robot. The sensitivity to external torques when the robot was not moving was further shown to be improved significantly by using the joint controllers with increased integral gain. The lead-through programming was implemented on two different robots and experimentally evaluated, and it was shown that the time for the programming phase can be substantially decreased compared to using the teach pendant. A version of this lead-through programming for ABB YuMi is now commercially available [ABB Robotics, 2015d].

11

Conclusions

This thesis presents work in the context of force-controlled robotic assembly. The focus has been on making it possible to specify tasks in a simple and intuitive way, and to include functionality to handle uncertainties and adaptation to different environments. The problem of replacing the force sensor with estimation of the contact forces using the internal sensors in the robot was also treated. Experiments were used throughout the thesis to validate the proposed methods.

A framework for robotic assembly was presented. An assembly task was composed of different skills, which both could be standard position-controlled procedures and skills using external sensing. A force-controlled skill was specified as a sequence of simple motions coordinated by a state machine, where each simple motion was specified using a constraint-based methodology. Transitions between the simple motions could either be triggered by threshold levels, or by more advanced classifiers based on machine learning. It was reported how the native position-based robot controller was integrated with an external controller performing force control. Two assembly scenarios were used to give detailed illustrations of the usage of the framework. The experimental implementations were compared to the performance of humans. Whereas the ability of the human to perform assembly is still superior to the robot, the robot can nonetheless be a competitive alternative by working around the clock. The major issue for using force-controlled assembly in an industrial setting are the requirements that then has to be fulfilled, usually including extreme high demands on uptime and robustness. For industrial usage, the number of times the robot stops due to error situations must be reduced, e.g., by making it easy to include detection of and recovery from errors.

It was described how uncertainties in assembly tasks can be managed. Geometric uncertainties were modeled by including uncertain transformations in the motion description, and the uncertainties were resolved by using sensor measurements. To illustrate the method, two different uncertainties were considered experimentally, namely an uncertain location

and a gripping uncertainty. Another type of uncertainty is the environmental properties. A method for adaptation of force control parameters based on identification of a contact model of the environment was presented. The method was experimentally shown to work for a wide range of environments with different stiffness properties. The ability of the assembly framework to handle uncertainties makes it easier for the robot operator to specify and accomplish assembly tasks.

Specification of an assembly task as was done in Chap. 3 is quite difficult and almost requires expert knowledge. Instructing a human how to perform an assembly task is, however, much simpler. It is usually enough to describe the final configuration, i.e., a high-level description is given. The thesis presents a method for how high-level specification of sensor-based skills can be performed for robots. The high-level description from the operator is used to generate an executable low-level description. Another method to simplify the task specification procedure is lead-through programming, i.e., to manually guide the robot. The thesis presents how this can be used to either teach important positions or entire trajectories for the robot.

Two methods for estimating the external forces applied to a robot without any force sensor were presented. The first one was based on the joint position-control errors. The method was simple to calibrate and implement, and it was good at detecting force transients, but the method was not especially good at estimating the correct force magnitude. The second method was based on the joint motor torques and modeling of the velocity-dependent friction uncertainty. This method had a lower estimation error than the first method, but a model for gravity compensation was needed and an optimization problem had to be solved numerically to get the force estimate. Both methods were implemented and were used to accomplish several assembly tasks although the resulting force estimate was not as good as a dedicated force sensor. A force sensor is usually quite expensive, and force estimation may therefore be a competitive alternative for many robotic tasks.

Friction in the joints is the major disturbance when performing force estimation. A method to increase the accuracy of force estimation using dithering was presented. A dithering signal was sent as a feedforward torque signal to decrease the Coulomb friction level when the joints were not moving. Experiments showed that the method made it possible to detect forces of magnitudes around 10 N, as compared to 20–30 N when no dithering was used.

Lead-through programming is useful for easy and intuitive robot programming. The thesis presents a method for sensorless lead-through programming based on disabling the low-level joint controllers and only feedforward the torque to balance gravity. As no position- or force-feedback

loops were active, the system remained stable in contact with any environment. The lead-through programming performance was improved by adding friction compensation. The approach was experimentally validated on two different industrial robots.

Concluding remarks

Force sensing provides robots the capability to accomplish assembly tasks in a robust way. Specifying the tasks is, however, a difficult problem, as well as tuning of all control parameters and transition conditions. The approach to use high-level task specification is promising, but it must be further developed to be useful in a real application. An alternative approach is programming by demonstration, i.e., to demonstrate the task and infer a task specification from the demonstration.

Achieving robust implementations of assembly tasks is tedious work. It includes detecting common error situations and designing strategies to either avoid them or recover from them. This process should at least be semi-automatic, i.e., the system should be able to learn from the error situations and be able to at least give suggestions to the operator for how to avoid the experienced errors in the future. Automatic detection and recovery may be possible by making the system aware of the desired and the current contact situation. By modeling possible contact situations in a contact graph, the system should be able to autonomously return to a desired contact situation.

In this thesis it was investigated how force control could be used in the context of assembly. In a real industrial setting, it may not always be the best option to resort to force-controlled solutions for every assembly operation. Position-controlled skills are easy and intuitive to program, and they should be used whenever possible. Their field of application can be expanded by using tools with passive compliance, such that some degree of position uncertainty can be handled. For some skills, however, force control is the best alternative, and for these cases it will be worthwhile to put down the extra effort into the task specification and programming phase.

Force estimation was shown to be an alternative to a force sensor in several assembly tasks. The accuracy of force estimation is usually good enough in many cases. Together with the reduced investment cost when a force sensor is not needed, force estimation expands the domain of tasks that can be robotized. For lead-through programming purposes, force estimation makes it possible to move the robot by applying forces anywhere on the robot. This can be compared to when a wrist-mounted force sensor is used, where only forces applied to the wrist of the robot

will result in movements.

The methods for force estimation presented in this thesis were focused on detecting small external forces, which were small in the sense that the interesting forces were in the same order of magnitude or smaller than the disturbances. Many of the previously published methods for force estimation have presented experiments where the signal-to-noise ratio was more favorable, and the resulting estimation problems thus simpler. Estimating small external forces was a challenging problem, which sometimes required modified strategies to be able to accomplish the assembly tasks.

Several methods to make specification easy and execution robust of assembly tasks have been presented in this thesis. The methods have, however, mostly been evaluated independently. It remains as future work to implement them all together to find out their full potential in the context of assembly.

A

Position-Control Reference Generator

The position controller in the acceleration level control scheme (Sec. 3.2) uses a reference generator to calculate the minimum-time trajectory to go from the current configuration to a desired position at rest. The trajectory is calculated for one coordinate at a time, i.e., the problem is one-dimensional. Both the velocity and acceleration should be bounded, but no limitations are put on the jerk. The resulting acceleration will be in the form of bang-bang control, according to

$$a(t) = \begin{cases} a^* & , \quad t_0 \leq t < t_1 \\ 0 & , \quad t_1 \leq t < t_2 \\ -a^* & , \quad t_2 \leq t < t_3 \end{cases} \tag{A.1}$$

where $|a^*| = a_{max}$ is the maximum acceleration with the sign to be calculated, the time t_0 is the start of the trajectory, and the times t_1, t_2, and t_3 should be calculated. The zero acceleration interval is needed if the maximum velocity, v_{max}, is attained. The trajectory generator is assumed to be invoked at time $t = 0$ with a reference position p_r together with the current position p_0 and current velocity v_0. If the start velocity $|v_0| > v_{max}$, an initial phase where the velocity is decreased is added. The acceleration profile now becomes

$$a(t) = \begin{cases} -\text{sign}(v_0)a_{max} & , \quad 0 \leq t < t_0 \\ a^* & , \quad t_0 \leq t < t_1 \\ 0 & , \quad t_1 \leq t < t_2 \\ -a^* & , \quad t_2 \leq t < t_3 \end{cases} \tag{A.2}$$

The trajectory is defined from the acceleration a^* and the times t_0, t_1,

t_2, and t_3. The velocity profile becomes

$$v(t) = \begin{cases} v_0 - \text{sign}(v_0)a_{max}t & , \quad 0 \leq t < t_0 \\ v(t_0) + a^*(t - t_0) & , \quad t_0 \leq t < t_1 \\ v(t_0) + a^*(t_1 - t_0) & , \quad t_1 \leq t < t_2 \\ v(t_0) + a^*(t_1 - t_0 - t + t_2) & , \quad t_2 \leq t < t_3 \end{cases} \tag{A.3}$$

where

$$t_0 = \begin{cases} \frac{|v_0| - v_{max}}{a_{max}} & , \quad |v_0| > v_{max} \\ 0 & , \quad \text{otherwise} \end{cases} \tag{A.4}$$

and

$$v(t_0) = v_0 - \text{sign}(v_0)a_{max}t_0 \tag{A.5}$$

The position profile becomes

$$p(t) = \begin{cases} p_0 + v_0 t - \frac{1}{2}\text{sign}(v_0)a_{max}t^2 & , \quad 0 \leq t < t_0 \\ p(t_0) + v(t_0)(t - t_0) + \frac{1}{2}a^*(t - t_0)^2 & , \quad t_0 \leq t < t_1 \\ p(t_1) + v(t_1)(t - t_1) & , \quad t_1 \leq t < t_2 \\ p(t_2) + v(t_2)(t - t_2) - \frac{1}{2}a^*(t - t_2)^2 & , \quad t_2 \leq t < t_3 \end{cases} \tag{A.6}$$

where

$$p(t_0) = p_0 + v_0 t_0 - \frac{1}{2}\text{sign}(v_0)a_{max}t_0^2 \tag{A.7}$$

and

$$p(t_1) = p(t_0) + v(t_0)(t_1 - t_0) + \frac{1}{2}a^*(t_1 - t_0)^2 \tag{A.8}$$

and

$$p(t_2) = p(t_1) + v(t_1)(t_2 - t_1) \tag{A.9}$$

Two cases are possible, depending on if the maximum velocity is attained or not. Firstly, case 1 below is tried, if no feasible solution is found, case 2 is used instead.

Case 1—Maximum velocity not attained The times for the trajectory are given as

$$t_1 = \frac{-v(t_0)}{a^*} \pm \sqrt{\frac{v(t_0)^2}{2a^{*2}} + \frac{p_r - p(t_0)}{a^*}} + t_0 \tag{A.10}$$

$$t_2 = t_1 \tag{A.11}$$

$$t_3 = \frac{-v(t_0)}{a^*} \pm 2\sqrt{\frac{v(t_0)^2}{2a^{*2}} + \frac{p_r - p(t_0)}{a^*}} + t_0 \tag{A.12}$$

Choose $a^* = a_{max}$ or $a^* = -a_{max}$ such that t_1, t_2, and t_3 becomes real and such that $t_3 \geq t_2 \geq t_1 \geq 0$. If multiple solutions exist, choose the one

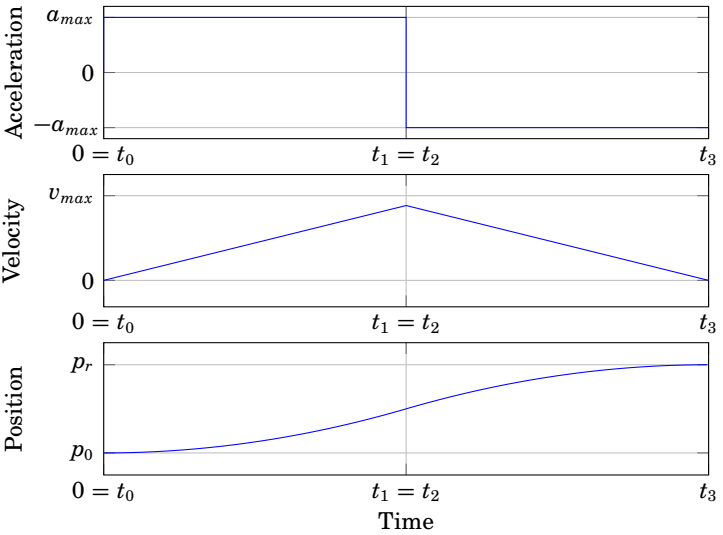


Figure A.1 A generated trajectory where the maximum velocity is not attained (Case 1).

where t_3 is the smallest. This solution is feasible if $|v_1| < v_{max}$, where v_1 is the largest velocity during the trajectory, given by

$$v_1 = a^*(t_3 - t_2) \tag{A.13}$$

An example of this type of trajectory is displayed in Fig. A.1.

Case 2—Maximum velocity attained The times for the trajectory are given as

$$t_1 = \frac{v^* - v(t_0)}{a^*} + t_0 \tag{A.14}$$

$$t_2 = t_0 + \frac{p_r - p(t_0)}{v^*} + \frac{v(t_0)^2}{2a^*v^*} - \frac{v(t_0)}{a^*} \tag{A.15}$$

$$t_3 = \frac{v^*}{a^*} + t_2 \tag{A.16}$$

where

$$v^* = \text{sign}(a^*)v_{max} \tag{A.17}$$

Choose $a^* = a_{max}$ or $a^* = -a_{max}$ such that $t_3 \geq t_2 \geq t_1 \geq 0$. If multiple solutions exist, choose the one where t_3 is the smallest. An example of this type of trajectory is displayed in Fig. A.2. In Fig. A.3, an example of a trajectory where the initial velocity, v_0, was too large is displayed.

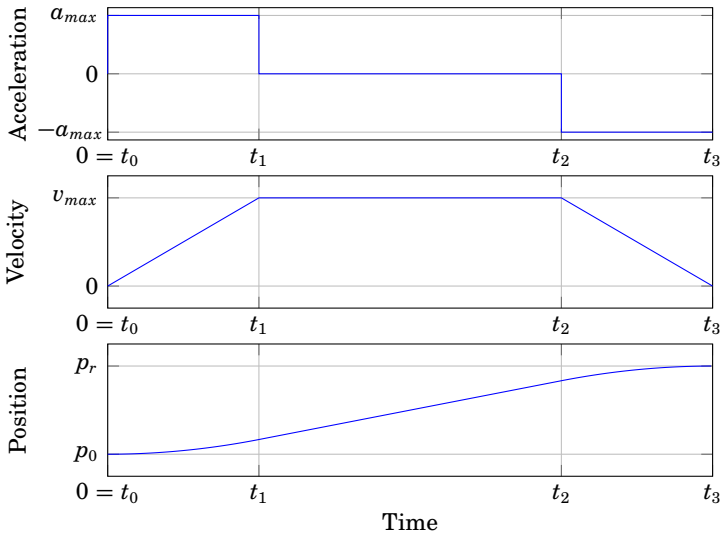


Figure A.2 A generated trajectory where the maximum velocity is attained (Case 2).

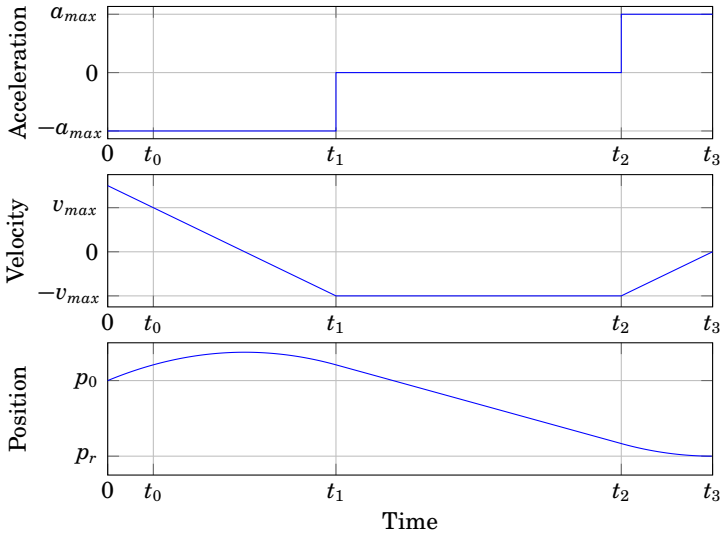


Figure A.3 A generated trajectory where the initial velocity was larger than the allowed maximum velocity v_{max}.

B

Derivation of Force Estimation Equations

In this chapter, the optimization problems (8.12) and (8.13) used to calculate the force estimates in the motor torque method in Sec. 8.3 are derived. The gravity compensated measured motor torque, $\bar{\tau} \in \mathbb{R}_n$, are modeled as

$$\bar{\tau} = J^T F + \tau_f + e \tag{B.1}$$

where n is the number of robot joints, $J \in \mathbb{R}_{n_F \times n}$ is the geometric robot Jacobian, $F \in \mathbb{R}_{n_F}$ is the force/torque applied at the end effector, $n_F = 6$ is the dimension of the force/torque vector, and e is a Gaussian disturbance according to

$$e \sim \mathcal{N}(0, R_e) \tag{B.2}$$

and τ_f is a uniformly distributed disturbance

$$\tau_f \sim \mathcal{U}(\tau_{f,min}, \tau_{f,max}) \tag{B.3}$$

and F is assumed to have a Gaussian prior distribution

$$F \sim \mathcal{N}(\bar{F}, R_F) \tag{B.4}$$

The probability density function for the total disturbance $\tau_f + e$ is

$$p_{\tau_f + e}(\tau_f, e) = p_{\tau_f}(\tau_f) p_e(e) \tag{B.5}$$

as τ_f and e are assumed to be independent, and

$$p_{\tau_f}(\tau_f) = \begin{cases} \frac{1}{\prod_{i=1}^{n} (\tau^i_{f,max} - \tau^i_{f,min})} & , \quad \tau_{f,min} \leq \tau_f \leq \tau_{f,max} \\ 0 & , \quad \text{otherwise} \end{cases} \tag{B.6}$$

where the superscript i refers to element i of the vector. Further, the probability density functions for e and F are given as

$$p_e(e) = \frac{1}{\sqrt{(2\pi)^n \det R_e}} \exp\left(-\frac{1}{2} e^T R_e^{-1} e\right) \tag{B.7}$$

$$p_F(F) = \frac{1}{\sqrt{(2\pi)^{n_F} \det R_F}} \exp\left(-\frac{1}{2}(F-\bar{F})^T R_F^{-1}(F-\bar{F})\right) \tag{B.8}$$

The likelihood function is the probability density function of the measurement $\bar{\tau}$ given the applied force/torque F

$$\begin{aligned} \mathcal{L}(F) &= p(\bar{\tau}|F) = p_{\tau_f+e}(\tau_f, \bar{\tau} - J^T F - \tau_f) \\ &= \begin{cases} \frac{\exp\left(-\frac{1}{2}(\bar{\tau}-J^T F-\tau_f)^T R_e^{-1}(\bar{\tau}-J^T F-\tau_f)\right)}{\prod_{i=1}^n \left(\tau^i_{f,max}-\tau^i_{f,min}\right)\sqrt{(2\pi)^n \det R_e}} & , \quad \tau_{f,min} \le \tau_f \le \tau_{f,max} \\ 0 & , \quad \text{otherwise} \end{cases} \end{aligned} \tag{B.9}$$

B.1 Maximum-likelihood

The maximum-likelihood (ML) estimate of F is the one that maximizes the likelihood function (B.9). Maximizing the likelihood function is equivalent to minimizing the negative loglikelihood, which is given as

$$-\log \mathcal{L}(F) = \begin{cases} \frac{(\bar{\tau}-J^T F-\tau_f)^T R_e^{-1}(\bar{\tau}-J^T F-\tau_f)}{2} + g_1(R_e) & , \quad \tau_{f,min} \le \tau_f \le \tau_{f,max} \\ \infty & , \quad \text{otherwise} \end{cases} \tag{B.10}$$

where $g_1(R_e)$ contains all terms not depending on F and τ_f. The force estimate, $\hat{F}$, is calculated by minimizing (B.10) as

$$\hat{F} = \operatorname{argmin}\left(-\log \mathcal{L}(F)\right) \tag{B.11}$$

Also τ_f is an optimization variable, and the minimization can be reformulated to the following optimization problem

$$\begin{aligned} &\underset{\text{over } F,\tau_f}{\text{minimize}} && \tfrac{1}{2}\left(\bar{\tau} - J^T F - \tau_f\right)^T R_e^{-1}\left(\bar{\tau} - J^T F - \tau_f\right) \\ &\text{subject to} && \tau_{f,min} \le \tau_f \le \tau_{f,max} \end{aligned} \tag{B.12}$$

where all terms not depending on F or τ_f have been excluded, and where a constraint has been added to account for the bounds of τ_f. The force estimate is the F solving (B.12).

B.2 Bayesian approach

In the Bayesian approach, the posterior is maximized instead of the likelihood [Murphy, 2012, p. 65–70]. This estimate is usually called the maximum a posteriori (MAP) estimate. The posterior $p(F|\bar{\tau})$ can be derived using Bayes law, according to

$$p(F|\bar{\tau}) = \frac{p(\bar{\tau}|F)p_F(F)}{\int p(\bar{\tau}|F)p_F(F)dF} = \frac{p(\bar{\tau}|F)p_F(F)}{p(\bar{\tau})} \tag{B.13}$$

In the same manner as with the ML-estimation, the negative logarithm of the posterior is minimized, which is given as

$$-\log p(F|\bar{\tau}) = \underbrace{-\log p(\bar{\tau}|F)}_{-\log \mathcal{L}(F)} - \log p_F(F) + \log p(\bar{\tau}) \tag{B.14}$$

The first term is the negative loglikelihood, the last term does not depend of F and can hence be excluded when performing the optimization. The middle term involving the prior, $p_F(F)$, is

$$-\log p_F(F) = \frac{(F-\bar{F})^T R_F^{-1}(F-\bar{F})}{2} + g_2(R_F) \tag{B.15}$$

where $g_2(R_F)$ contains all terms not depending on F. The minimization of the posterior results in the same optimization problem as for the ML-estimation (B.12), but with the difference that the first term of (B.15) is added to the objective function, according to

$$\begin{aligned} \underset{\text{over } F,\tau_f}{\text{minimize}} \quad & \tfrac{1}{2}\left(\bar{\tau} - J^T F - \tau_f\right)^T R_e^{-1}\left(\bar{\tau} - J^T F - \tau_f\right) \\ & + \tfrac{1}{2}\left(F-\bar{F}\right)^T R_F^{-1}\left(F-\bar{F}\right) \\ \text{subject to} \quad & \tau_{f,min} \leq \tau_f \leq \tau_{f,max} \end{aligned} \tag{B.16}$$

Bibliography

ABB Robotics (2011). *Application manual - SoftMove*. Document ID: 3HAC030266-001.

ABB Robotics (2012). *RAPID Reference Manual*. Document ID: 3HAC029364-001.

ABB Robotics (2015a). *ABB IRB120*. URL: http://new.abb.com/products/robotics/industrial-robots/irb-120/. Last visited: 2015-09.

ABB Robotics (2015b). *ABB IRB140*. URL: http://new.abb.com/products/robotics/industrial-robots/irb-140/. Last visited: 2015-09.

ABB Robotics (2015c). *ABB RobotStudio*. URL: http://new.abb.com/products/robotics/robotstudio/. Last visited: 2015-09.

ABB Robotics (2015d). *ABB YuMi*. URL: http://new.abb.com/products/robotics/yumi/. Last visited: 2015-09.

Albu-Schäffer, A., S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger (2007). "The DLR lightweight robot: design and control concepts for robots in human environments". *Industrial Robot: An International Journal* **34**:5, pp. 376–385.

Alcocer, A., A. Robertsson, A. Valera, and R. Johansson (2003). "Force estimation and control in robot manipulators". In: *Proc. IFAC Symp. Robot Control (SYROCO 2003)*. Wrocław, Poland, pp. 31–36.

An, C. and J. Hollerbach (1987). "Dynamic stability issues in force control of manipulators". In: *Proc. American Control Conf. (ACC 1987)*. Minneapolis, MN, USA, pp. 821–827.

Ang, V., L. Wei, and S. Y. Lim (2000). "An industrial application of control of dynamic behavior of robots - a walk-through programmed welding robot". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2000)*. Vol. 3. San Francisco, CA, USA, pp. 2352–2357.

Arai, T., N. Yamanobe, Y. Maeda, H. Fujii, T. Kato, and T. Sato (2006). "Increasing Efficiency of Force-Controlled Robotic Assembly -Design of Damping Control Parameters Considering Cycle Time". *CIRP Annals-Manufacturing Technology* **55**:1, pp. 7–10.

Åström, K. J. and B. Wittenmark (1996). *Computer-controlled systems: theory and design*. Prentice Hall, New York, NY, USA.

ATI (2015). URL: http://www.ati-ia.com/products/ft/sensors.aspx. Last visited: 2015-09.

Bagge Carlson, F., A. Robertsson, and R. Johansson (2015). "Modeling and identification of position and temperature dependent friction phenomena without temperature sensing". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2015)*. Hamburg, Germany.

Beetz, M., L. Mösenlechner, and M. Tenorth (2010). "CRAM–A cognitive robot abstract machine for everyday manipulation in human environments". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2010)*. Taipei, Taiwan, pp. 1012–1017.

Bishop, C. (2006). *Pattern recognition and machine learning*. Springer, New York, NY, USA.

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: *Proc. ICRA 2010 Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications*. Anchorage, AK, USA, pp. 62–66.

Bruyninckx, H. and J. De Schutter (1996). "Specification of force-controlled actions in the 'task frame formalism'-a synthesis". *IEEE Transactions on Robotics and Automation* **12**:4, pp. 581–589.

Bruyninckx, H., T. Lefebvre, L. Mihaylova, E. Staffetti, J. De Schutter, and J. Xiao (2001). "A roadmap for autonomous robotic assembly". In: *Proc. Int. Symp. on Assembly and Task Planning*. Fukuoka, Japan.

Buerger, S. and N. Hogan (2006). "Relaxing passivity for human-robot interaction". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2006)*. Beijing, China, pp. 4570–4575.

Caccavale, F., C. Natale, B. Siciliano, and L. Villani (1998). "Control of two industrial robots for parts mating". In: *Proc. Int. Conf. Control Applications (CCA 1998)*. Vol. 1. IEEE. Trieste, Italy, pp. 562–566.

Calinon, S. and A. Billard (2007). "Active teaching in robot programming by demonstration". In: *Proc. IEEE Int. Symp. Robot and Human Interactive Communication (RO-MAN 2007)*. Jeju Island, Korea, pp. 702–707.

Chen, Z. and P. Kazanzides (2013). "Force control of a non-backdrivable robot without a force sensor". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2013)*. Tokyo, Japan, pp. 3570–3575.

Chiaverini, S. and L. Sciavicco (1993). "The parallel approach to force/position control of robotic manipulators". *IEEE Transactions on Robotics and Automation* **9**:4, pp. 361–373.

Cho, H., M. Kim, H. Lim, and D. Kim (2014). "Cartesian sensor-less force control for industrial robots". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2014)*. Chicago, IL, USA, pp. 4497–4502.

Cho, S., S. Asfour, A. Onar, and N. Kaundinya (2005). "Tool breakage detection using support vector machine learning in a milling process". *International Journal of Machine Tools and Manufacture* **45**:3, pp. 241–249.

CVX Research, I. (2012). *CVX: matlab software for disciplined convex programming, version 2.0*. URL: http://cvxr.com/cvx/.

Dalal, N. and B. Triggs (2005). "Histograms of oriented gradients for human detection". In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2005)*. Vol. 1. San Diego, CA, USA, pp. 886–893.

De Luca, A., A. Albu-Schäffer, S. Haddadin, and G. Hirzinger (2006). "Collision detection and safe reaction with the DLR-III lightweight manipulator arm". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2006)*. Beijing, China, pp. 1623–1630.

De Luca, A. and R. Mattone (2005). "Sensorless robot collision detection and hybrid force/motion control". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2005)*. Barcelona, Spain, pp. 999–1004.

De Schutter, J. (1988). "Improved force control laws for advanced tracking applications". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 1988)*. Philadelphia, PA, USA, pp. 1497–1502.

De Schutter, J., T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx (2007). "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty". *The International Journal of Robotics Research* **26**:5, pp. 433–455.

Di Lello, E., T. De Laet, and H. Bruyninckx (2012). "Hierarchical Dirichlet Process Hidden Markov models for abnormality detection in robotic assembly". In: *Proc. NIPS 2012 Workshop on Bayesian Nonparametric*

Models (BNPM) For Reliable Planning And Decision-Making Under Uncertainty. Lake Tahoe, NV, USA.

Donald, B. (1986). "Robot motion planning with uncertainty in the geometric models of the robot and environment: a formal framework for error detection and recovery". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 1986)*. Vol. 3. San Francisco, CA, USA, pp. 1588–1593.

Du, H. and S. Nair (1999). "Modeling and compensation of low-velocity friction with bounds". *IEEE Transactions on Control Systems Technology* **7**:1, pp. 110–121.

Eom, K., I. Suh, W. Chung, and S. Oh (1998). "Disturbance observer based force control of robot manipulator without force sensor". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 1998)*. Leuven, Belgium, pp. 3012–3017.

Erickson, D., M. Weber, and I. Sharf (2003). "Contact stiffness and damping estimation for robotic systems". *The International Journal of Robotics Research* **22**:1, pp. 41–57.

Freidovich, L., A. Shiriaev, A. Robertsson, and R. Johansson (2010). "LuGre-model-based friction compensation". *IEEE Transactions on Control Systems Technology* **18**:1, pp. 194–200.

Freund, Y. and R. Schapire (1996). "Experiments with a new boosting algorithm". In: *Proc. Thirteenth Int. Conf. Machine Learning*. Bari, Italy, pp. 148–156.

Gambao, E., C. Balaguer, and F. Gebhart (2000). "Robot assembly system for computer-integrated construction". *Automation in Construction* **9**:5-6, pp. 479–487.

Gill, A. (1962). *Introduction to the Theory of Finite-state Machines*. McGraw-Hill, New York, NY, USA.

Grant, M. and S. Boyd (2008). "Graph implementations for nonsmooth convex programs". In: Blondel, V. et al. (Eds.). *Recent Advances in Learning and Control*. Lecture Notes in Control and Information Sciences. URL: http://stanford.edu/~boyd/graph_dcp.html. Springer-Verlag Limited, London, United Kingdom, pp. 95–110.

Gravel, D., F. Maslar, G. Zhang, S. Nidamarthi, H. Chen, and T. Fuhlbrigge (2008). "Toward robotizing powertrain assembly". In: *7th World Congress Int. Control and Automation*. Chongqing, China, pp. 541–546.

Hacksel, P. and S. Salcudean (1994). "Estimation of environment forces and rigid-body velocities using observers". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 1994)*. San Diego, CA, USA, pp. 931–936.

Haddadi, A. and K. Hashtrudi-Zaad (2008). "Online contact impedance identification for robotic systems". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2008)*. Nice, France, pp. 974–980.

Hägglund, T. (2002). "A friction compensator for pneumatic control valves". *Journal of Process Control* **12**:8, pp. 897–904.

Hogan, N. (1985). "Impedance control: an approach to manipulation: parts i-iii". *Journal of Dynamic Systems, Measurement, and Control* **107**, pp. 1–24.

Hsueh, Y. and C. Yang (2008). "Prediction of tool breakage in face milling using support vector machine". *The International Journal of Advanced Manufacturing Technology* **37**:9, pp. 872–880.

Huang, S. and J. M. Schimmels (2003). "Sufficient conditions used in admittance selection for force-guided assembly of polygonal parts". *IEEE Transactions on Robotics and Automation* **19**:4, pp. 737–742.

Huckaby, J. and H. Christensen (2012). "A taxonomic framework for task modeling and knowledge transfer in manufacturing robotics". In: *Proc. 26th AAAI Cognitive Robotics Workshop*. Toronto, Canada, pp. 94–101.

Ipri, S. and H. Asada (1995). "Tuned dither for friction suppression during force-guided robotic assembly". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 1995)*. Pittsburgh, PA, USA, pp. 310–315.

JGrafchart (2015). URL: http://www.control.lth.se/Research/tools/grafchart.html. Last visited: 2015-09.

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

Jörg, S., J. Langwald, C. Natale, J. Stelter, and G. Hirzinger (2000). "Flexible robot-assembly using a multi-sensory approach". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2000)*. San Francisco, CA, USA, pp. 3687–3694.

JR3 (2015). URL: http://www.jr3.com/. Last visited: 2015-09.

Jung, J., J. Lee, and K. Huh (2006). "Robust contact force estimation for robot manipulators in three-dimensional space". *Proc. Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* **220**:9, pp. 1317–1327.

Kailath, T., A. Sayed, and B. Hassibi (2000). "Linear estimation". *Prentice Hall, Upper Saddle River, NJ*, pp. 95–98.

Kalman, R. (1960). "A new approach to linear filtering and prediction problems". *Transactions of the ASME–Journal of Basic Engineering* **82**:Series D, pp. 35–45.

Kawada Industries (2015). *Kawada Nextage*. URL: http : / / nextage . kawada.jp/en/. Last visited: 2015-09.

Khatib, O. (1987). "A unified approach for motion and force control of robot manipulators: the operational space formulation". *IEEE Journal on Robotics and Automation* **3**:1, pp. 43–53.

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: *Proc. IEEE Int. Symp. Assembly and Manufacturing (ISAM 2011)*. Tampere, Finland, pp. 1–5.

Krebs, H. I., M. Ferraro, S. P. Buerger, M. J. Newbery, A. Makiyama, M. Sandmann, D. Lynch, B. T. Volpe, and N. Hogan (2004). "Rehabilitation robotics: pilot trial of a spatial extension for MIT-Manus". *Journal of NeuroEngineering and Rehabilitation* **1**:5.

Kresse, I. and M. Beetz (2012). "Movement-aware action control–Integrating symbolic and control-theoretic action execution". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2012)*. St Paul, MN, USA, pp. 3245–3251.

Kröger, T., B. Finkemeyer, M. Heuck, and F. Wahl (2004). "Adaptive implicit hybrid force/pose control of industrial manipulators: compliant motion experiments". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2004)*. Sendai, Japan, pp. 816–821.

Kröger, T., B. Finkemeyer, and F. Wahl (2011). "Manipulation primitives–A universal interface between sensor-based motion control and robot programming". In: Schütz, D. et al. (Eds.). *Robotic Systems for Handling and Assembly*. Springer, Berlin, Heidelberg, Germany, pp. 293–313.

LabComm (2015). URL: http://www.control.lth.se/Research/tools.html. Last visited: 2015-09.

Lee, S. and H. Asada (1995). "Direct adaptive control of force-guided assembly robots using tuned dither". In: *Proc. American Control Conf. (ACC 1995)*. Seattle, WA, USA, pp. 370–374.

Lefebvre, T., H. Bruyninckx, and J. De Schutter (2005). "Task planning with active sensing for autonomous compliant motion". *The International Journal of Robotics Research* **24**:1, pp. 61–81.

Lipshitz, S., R. Wannamaker, and J. Vanderkooy (1992). "Quantization and dither: a theoretical survey". *Journal of the Audio Engineering Society* **40**:5, pp. 355–375.

Love, L. and W. Book (1995). "Environment estimation for enhanced impedance control". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 1995)*. Nagoya, Japan, pp. 1854–1859.

Lundberg, I. (2013). Personal communication.

Lunze, J. and F. Lamnabhi-Lagarrigue, (Eds.) (2009). *Handbook of hybrid systems control: theory, tools, applications*. Cambridge University Press, Cambridge, U.K.

Mallapragada, V., D. Erol, and N. Sarkar (2006). "A new method of force control for unknown environments". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2006)*. Beijing, China, pp. 4509–4514.

Mason, M. T. (1981). "Compliance and force control for computer controlled manipulators". *IEEE Transactions on Systems, Man and Cybernetics* **11**:6, pp. 418–432.

Mathworks (2015a). *Simulink*. URL: http://www.mathworks.com/products/simulink/. Last visited: 2015-09.

Mathworks (2015b). *Simulink Coder*. URL: http://www.mathworks.com/products/simulink-coder/. Last visited: 2015-09.

Mathworks (2015c). *Stateflow*. URL: http://www.mathworks.com/products/stateflow/. Last visited: 2015-09.

Matthias, B., S. Kock, H. Jerregård, M. Källman, I. Lundberg, and R. Mellander (2011). "Safety of collaborative industrial robots: certification possibilities for a collaborative assembly robot concept". In: *Proc. IEEE Int. Symp. Assembly and Manufacturing (ISAM 2011)*. 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.

Mitsi, S., K.-D. Bouzakis, G. Mansour, D. Sagris, and G. Maliaris (2005). "Off-line programming of an industrial robot for manufacturing". *The International Journal of Advanced Manufacturing Technology* **26**:3, pp. 262–267.

Murakami, T., F. Yu, and K. Ohnishi (1993). "Torque Sensorless Control in Multidegree-of-Freedom Manipulator". *IEEE Transactions on Industrial Electronics* **40**:2, pp. 259–265.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press, Cambridge, MA, USA.

Natale, C., R. Koeppe, and G. Hirzinger (2000). "A systematic design procedure of force controllers for industrial robots". *IEEE/ASME Transactions on Mechatronics* **5**:2, pp. 122–131.

Ohishi, K. (1993). "Sensorless force control using H^{∞} acceleration controller". In: *Proc. Asia-Pacific Workshop on Advances in Motion Control*. Singapore, pp. 13–18.

Ohishi, K., M. Miyazaki, and M. Fujita (1992). "Hybrid control of force and position without force sensor". In: *Proc. Int. Conf. Industrial Electronics, Control, Instrumentation, and Automation, Power Electronics and Motion Control*. San Diego, CA, USA, pp. 670–675.

Ohishi, K., M. Miyazaki, M. Fujita, and Y. Ogino (1991). "H^{∞} observer based force control without force sensor". In: *Proc. Int. Conf. Industrial Electronics, Control and Instrumentation*. Kobe, Japan, pp. 1049–1054.

Olsson, H., K. Åström, C. Canudas de Wit, M. Gäfvert, and P. Lischinsky (1998). "Friction models and friction compensation". *European Journal of Control* **4**:3, pp. 176–195.

Olsson, T., M. Haage, H. Kihlman, R. Johansson, K. Nilsson, A. Robertsson, M. Björkman, R. Isaksson, and G. Ossbahr (2010). "Cost-efficient drilling using industrial robots with high-bandwidth force feedback". *Robotics and Computer-Integrated Manufacturing* **26**:1, pp. 24–38.

OROCOS (2015). URL: http://www.orocos.org/wiki/orocos/itasc-wiki/. Last visited 2015-09.

Pan, Z., J. Polden, N. Larkin, S. Van Duin, and J. Norrish (2012). "Recent progress on programming methods for industrial robots". *Robotics and Computer-Integrated Manufacturing* **28**:2, pp. 87–94.

Park, S. and S. Lee (2007). "Disturbance observer based robust control for industrial robots with flexible joints". In: *Proc. Int. Conf. Control, Automation and Systems (ICCAS 2007)*. IEEE. Seoul, Korea, pp. 584–589.

Pedersen, M. R., D. L. Herzog, and V. Krüger (2014). "Intuitive skill-level programming of industrial handling tasks on a mobile manipulator". In: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2014)*. Chicago, IL, USA, pp. 4523–4530.

Popovic, M. and A. Goldenberger (1998). "Modeling of Friction Using Spectral Analysis". *IEEE Transactions on Robotics and Automation* **14**:1, pp. 114–122.

Raibert, M. and J. Craig (1981). "Hybrid position/force control of manipulators". *Journal of Dynamic Systems, Measurement, and Control* **102**:127, pp. 126–133.

Rethink Robotics (2015). *Baxter*. URL: http://www.rethinkrobotics.com/baxter/. Last visited: 2015-09.

Rodriguez, A., D. Bourne, M. Mason, G. Rossano, and J. Wang (2010). "Failure detection in assembly: force signature analysis". In: *Proc. IEEE Int. Conf. Automation Science and Engineering (CASE 2010)*. 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: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2012)*. Vilamoura, Portugal, pp. 356–363.

ROS (2015). *Robot Operating System*. URL: http://www.ROS.org/. Last visited: 2015-09.

Roy, J. and L. Whitcomb (2002). "Adaptive force control of position/velocity controlled robots: theory and experiment". *IEEE Transactions on Robotics and Automation* **18**:2, pp. 121–137.

Sararoody, M., F. Sheikholeslam, and M. Keshmiri (2005). "A force estimator based algorithm for robot control". In: *Proc. IEEE Int. Conf. Mechatronics (ICM 2005)*. Taipei, Taiwan, pp. 376–381.

Siciliano, B., L. Sciavicco, L. Villani, and G. Oriolo (2009). *Robotics: modelling, planning and control*. Springer-Verlag, London, U.K., pp. 259–264.

Simpson, J., C. Cook, and Z. Li (2002). "Sensorless Force Estimation for Robots with Friction". In: *Proc. Australasian Conf. Robotics and Automation*. Auckland, New Zealand, pp. 94–99.

Smith, O. (1957). "Closer control of loops with dead time". *Chemical Engineering Progress* **53**:5, pp. 217–219.

Smits, R. (2010). *Robot skills: design of a constraint-based methodology and software support*. PhD thesis. Dept. Mech. Eng., KU Leuven, Belgium.

Stenmark, M. and J. Malec (2014). "Describing constraint-based assembly tasks in unstructured natural language". In: *Proc. IFAC 2014 World Congress*. Cape Town, South Africa, pp. 3056–3061.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2012). "Robotic assembly using a singularity-free orientation representation based on quaternions". In: *Proc. IFAC Symp. Robot Control (SYROCO 2012)*. Dubrovnik, Croatia, pp. 549–554.

Tachi, S., T. Sakaki, H. Arai, S. Nishizawa, and J. Pelaez-Polo (1990). "Impedance control of a direct-drive manipulator without using force sensors". *Advanced Robotics* **5**:2, pp. 183–205.

Thrun, S. (2002). "Probabilistic robotics". *Communications of the ACM* **45**:3, pp. 52–57.

Toshiba (2015). *Sensor-less compliance control for assembly robots*. URL: http://www.toshiba-machine.co.jp/en/technology/tech_catalog/e3.html. Last visited: 2015-09.

Van Damme, M., B. Beyl, V. Vanderborght, V. Grosu, R. Van Ham, I. Vanderniepen, A. Matthys, and D. Lefeber (2011). "Estimating Robot End-Effector Force from Noisy Actuator Torque Measurements". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2011)*. Shanghai, China, pp. 1108–1113.

Verscheure, D., I. Sharf, H. Bruyninckx, J. Swevers, and J. De Schutter (2010). "Identification of contact parameters from stiff multi-point contact robotic operations". *The International Journal of Robotics Research* **29**:4, pp. 367–385.

Villani, L., C. C. De Wit, and B. Brogliato (1999). "An exponentially stable adaptive control for force and position tracking of robot manipulators". *IEEE Transactions on Automatic Control* **44**:4, pp. 798–802.

Wahrburg, A., S. Zeiss, B. Matthias, and H. Ding (2014). "Contact force estimation for robotic assembly using motor torques". In: *Proc. IEEE Int. Conf. Automation Science and Engineering (CASE 2014)*. Taipei, Taiwan, pp. 1252–1257.

Wahrburg, A., S. Zeiss, B. Matthias, and H. Ding (2015). "Cartesian contact force estimation for robotic manipulators using kalman filters and the generalized momentum". In: *Proc. IEEE Int. Conf. Automation Science and Engineering (CASE 2015)*. Gothenburg, Sweden, pp. 1230–1235.

Waibel, M., M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Gálvez-López, K. Häussermann, R. Janssen, J. Montiel, A. Perzylo, B. Schießle, M. Tenorth, O. Zweigle, and R. van de Molengraft (2011). "RoboEarth". *IEEE Robotics & Automation Magazine* **18**:2, pp. 69–82.

Weber, E., K. Patel, O. Ma, and I. Sharf (2006). "Identification of contact dynamics model parameters from constrained robotic operations". *Journal of Dynamic Systems, Measurement, and Control* **128**, pp. 307–318.

Willow Garage (2015). *PR2*. URL: http://www.willowgarage.com/pages/pr2/overview/. Last visited: 2015-09.

Wrede, S., C. Emmerich, R. Grünberg, A. Nordmann, A. Swadzba, and J. Steil (2013). "A user study on kinesthetic teaching of redundant robots in task and configuration space". *Journal of Human-Robot Interaction* **2**:1, pp. 56–81.

Xenomai (2015). URL: http://www.xenomai.org/. Last visited: 2015-09.

Xiao, J. and R. Volz (1989). "On replanning for assembly tasks using robots in the presence of uncertainties". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 1989)*. Vol. 2. Scottsdale, AZ, USA, pp. 638–645.

Zhu, H. and H. Fujimoto (2013). “Overcoming current quantization effects for precise current control using dithering techniques”. *IEEJ Journal of Industry Applications* **2**:1, pp. 14–21.