Robotic Assembly and Contact Force Control

Stolt, Andreas

2012

Document Version:
Publisher's PDF, also known as Version of record

Robotic Assembly and Contact Force Control

Andreas Stolt

Department of Automatic Control
Lund University
Lund, December 2012

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

ISSN 0280–5316
ISRN LUTFD2/TFRT--3256--SE

Printed in Sweden,
Lund University, Lund 2012

Abstract

Modern industrial robots are traditionally programmed to follow desired trajectories, with the only feedback coming from the internal position/angle sensors in the joints. The robots are in general very accurate in tracking the desired motion, and they have become indispensable in many applications, such as spot welding and painting in the automotive industry. In more complex tasks, such as physical interaction with the environment, position control of the robot might be insufficient due to the fact that it is hard, or too costly, to achieve an environment that is structured enough. This is due to inherent uncertainties, such as part variations and inexact gripping.

One example of a challenging application is assembly, which is hard to accomplish using only position controlled robots. By adding a force sensor to the system, it gives the robot ability to correct for uncertainties by measuring contacts. This thesis presents a framework for force controlled robotic assembly. Assembly tasks are specified as sequences of constrained motions, where transitions are triggered by sensor events, coming either from thresholds or from more advanced classifiers. The framework is also able to explicitly deal with uncertainties, which can be estimated during execution to improve the performance. Further, a method for adaptation of force control parameters is presented, and how a singularity-free orientation representation can be used within the assembly framework. The case when no force sensor is available is also considered, and a method for estimating the external forces based on the joint control errors is presented. All methods presented are validated in experiments.

Acknowledgments

First of all I would like to 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. 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 also want to thank the rest of my colleagues in the ROSETTA project for many interesting discussions, collaborations, and feedback on my work.

The Department of Automatic Control offers a very nice environment to work in, and I am very thankful for being a part of this. Administrative matters always run smoothly with the help from Eva Westin, Ingrid Nilsson, Lizette Borgeram, and Monika Rasmusson. The computers are almost always working as intended, thanks to Anders Nilsson, Anders Blomdell, and Leif Andersson for this. For all practical issues in the lab, Rolf Braun and Pontus Andersson are always willing to help.

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”, an honorary title awarded at a summer school in Bavaria 2011, together with Magnus Linderoth, Olof Sörnmo, Björn Olofsson,

and Karl Berntorp.

I would also like to thank Olof Sörnmo, Martin Hast, Josefin Berner, and Meike Stemman for sharing office with me. You have all contributed for creating a good working environment, where every possible issue can be discussed.

Finally I would like to thank my wife Emma, my family, and friends.

The work has been conducted within the LCCC Linnaeus Center (supported by the Swedish Research Council) and within the ELLIIT strategic research area.

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.

Contents

1

Introduction

1.1 Motivation and background

The traditional way of programming industrial robots is using position control, i.e., the task is to follow desired trajectories, and the only sensors used are usually the position sensors in the joints. Modern robot controllers are very good at these tasks and are able to perform them very fast and with good repetitive accuracy. Typical applications include welding, painting, packaging, and palletizing. The introduction of robots in industry has relieved human workers from repetitive and/or dangerous tasks, where the automotive industry is one example.

Assembly is an example of an application that is less robotized. This kind of application contains different sources of uncertainty that the robot systems need to be able to handle, such as part variations and inaccurate gripping, and the tasks often have small tolerances. Doing assembly with position controlled robots is possible, but it requires a very good accuracy of the workcell. This can be achieved by using different kinds of fixtures and task specific solutions. This method is, however, quite inflexible, e.g., a small change in the task may make the previously used fixtures unusable, and new ones has to be designed, which may require a lot of work, and even more work is required if also the robot control program has to be changed.

The current trend in robotics seems to be going from the stiff and

very accurate traditional industrial robots, to mobile and more light-weight robots, with the PR2 [Willow Garage, 2012] and the Baxter robot [Rethink Robotics, 2012] as recent examples. In this way, the possibility of using robots close to humans opens up, as the robots no longer are as dangerous as they used to be, with compliant structures, reduced mass, and less power. Another example of a recent light-weight robot is the ABB concept robot FRIDA [Kock *et al.*, 2011], which is one of the experimental platforms in this thesis. This new kind of robots is weaker than traditional ones, but this can be advantageous in tasks such as assembly. The robots will be less prone to break parts because of the compliance, but this feature will on the other hand introduce more uncertainties, which will have to be taken care of.

Introducing additional sensing is a way to compensate for an insufficient position accuracy, e.g., caused by tasks with small tolerances, or inaccurate robots. A force sensor gives the robot capabilities to be used in tasks where physical interaction with the involved objects is central, as position uncertainties can be corrected for by sensing the contact forces. However, additional sensing makes the specification of tasks a harder problem, more advanced than just to follow a desired trajectory. It should be possible for non-expert robot users to specify advanced sensor-based tasks for the success of robotics to continue.

In this thesis, the research field of force controlled robotic assembly is considered. The problem of specifying tasks and executing them is a central part, but adaptation to uncertainties and different environments is also considered. Experiments are used throughout the thesis to show the usefulness of the contributions.

1.2 Contributions

The thesis contains the following contributions

- a framework for general force-controlled assembly tasks, including task specification and uncertainty management;
- a method for self-tuning of force controllers used in robots, and integration of the method in a framework for performing assembly;

- a method for using a singularity-free orientation representation in an assembly framework;
- a method for estimating external forces acting on the end-effector of a robot, based on the joint control errors.

1.3 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. 2011 IEEE International Conference on Robotics and Automation (ICRA2011)*. Shanghai, China, May 9–13, 2011, pp. 3751–3756.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2012): "Force Controlled Robotic Assembly without a Force Sensor" In *Proc. 2012 IEEE International Conference on Robotics and Automation (ICRA2012)*. St Paul, Minnesota, USA, May 14–18, 2012, pp. 1538–1543. (Best automation paper award)

The above publications have mainly been written by the author together with M. Linderoth, but most of the parts included in this thesis have been made mostly by the author. For the work on force estimation, however, equal contribution is asserted. A. Robertsson and R. Johansson mainly assisted with structuring the manuscripts and discussing the ideas.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2012): "Adaptation of Force Control Parameters in Robotic Assembly" In *Proc. 2012 IFAC Symposium on Robot Control (SYROCO2012)*. Dubrovnik, Croatia, September 5–7, 2012, pp. 561–566.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2012): "Robotic Assembly Using a Singularity-Free Orientation Representation Based on Quaternions " In *Proc. 2012 IFAC Symposium on Robot Control (SYROCO2012)*. Dubrovnik, Croatia, September 5–7, 2012, pp. 549–554.

The above publications have the author as main contributor. M. Linderoth assisted in developing the ideas, and A. Robertsson and R. Johansson assisted on structuring the manuscripts.

The assembly framework used in all of the above publications has been developed in close cooperation between the author and M. Linderoth.

Other publications

Björkelund, A., L. Edström, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. Gestegård Robertz, D. Störkle, A. Blomdell, R. Johansson, M. Linderoth, A. Nilsson, A. Robertsson, A. Stolt and H. Bruyninckx (2011): "On the Integration of Skilled Robot Motions for Productivity in Manufacturing" In *Proc. 2011 IEEE/CIRP International Symposium on Assembly and Manufacturing (ISAM2011)*. Tampere, Finland, May 25–27, 2011, pp. 1–9.

The above publication is considering the same assembly scenario as 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. 2010 SAE Aerospace Manufacturing and Automated Fastening Conference*. Wichita, Kansas, USA, September 28–30, 2010.

Stolt, A., M. Linderoth, A. Robertsson, M. Jonsson, and T. Murray (2011): "Force Controlled Assembly of Flexible Aircraft Structure" In *Proc. 2011 IEEE International Conference on Robotics and Automation (ICRA2011)*. Shanghai, China, May 9–13, 2011, pp. 6027–6032.

Jonsson, M., A. Stolt, A., Robertsson, T. Murray, and K. Nilsson (2011): "Force Controlled Assembly of a Compliant Rib" In *Proc. SAE 2011 Aerotech Congress & Exhibition*. Toulouse, France, October 18–21, 2011.

Jonsson, M., A. Stolt, A., Robertsson, S. von Gegerfelt, and K. Nilsson (2012): "On force control for assembly and cleaning of castings"

Submitted to *The International Journal of Advanced Manufacturing Technology*.

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.

1.4 Thesis outline

Chapter 2 describes the robots, other hardware and interfaces used throughout the thesis. Chapter 3 presents the emergency stop button assembly scenario, which is used as an example in large parts of the thesis. In Chapter 4, the assembly framework is described, i.e., how tasks are modeled and control is performed. The implementation of an assembly task using the control framework is presented in Chapter 5, together with how a machine learning approach was used for detecting when the assembly was finished. The topic of Chapter 6 is how uncertainties can be modeled and resolved within the assembly framework.

Chapter 7 presents a method for adaptation of force control parameters and how it is integrated with the assembly framework. In Chapter 8, it is described how a singularity-free orientation representation based on quaternions can be used in the assembly framework. A method for estimation of external forces together with two examples of assembly scenarios where the method is used is presented in Chapter 9. Finally, conclusions are given in Chapter 10.

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. The following section gives a brief overview of the hardware used.

2.1 Robots

Two different robots have been used in this thesis. The first one is the ABB IRB140 robot [ABB Robots, 2012] (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 used was ABB FRIDA [Kock *et al.*, 2011] (see Fig. 2.2), a dual arm concept robot not yet in production. 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.

Both of the robots are controlled by the ABB IRC5 industrial control system.

2.2 Robot controller interface

The industrial control system, ABB IRC5, has been extended with an open robot control system [Blomdell *et al.*, 2005; Blomdell *et al.*, 2010]. This makes it possible to connect to the low-level joint controllers, and modify the references for joint positions and feed-forward of velocity and motor torque. Measurements available are, e.g., measured joint positions and motor torques. The sampling rate available is 250 [Hz].

The external controller is executed on an external PC running with Linux and Xenomai [Xenomai, 2012] for real-time performance. The communication with the robot controller is made with the LabComm protocol [LabComm, 2012], 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.

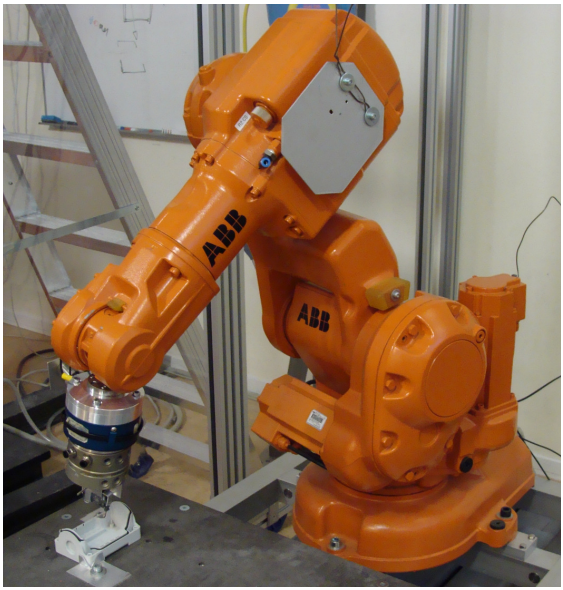

Figure 2.1 The ABB IRB140 robot used in the experiments in the thesis. The robot is equipped with a JR3 force/torque sensor.

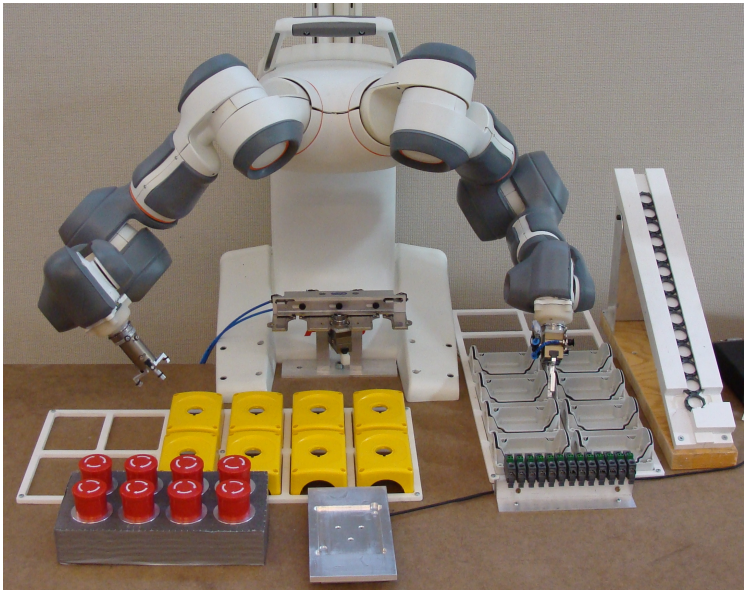

Figure 2.2 The ABB FRIDA robot used in the experiments in the thesis.

2.3 Force sensing

The IRB140 robot was equipped with a wrist-mounted 6 degrees-of-freedom JR3 100M40A force/torque sensor [JR3, 2012]. The workcell with the FRIDA robot was equipped with a table-mounted 6 degrees-of-freedom ATI Mini40 force/torque sensor [ATI, 2012].

3

The emergency stop button assembly scenario

An assembly scenario that will be used as an example throughout this thesis is the assembly of an emergency stop button. The scenario is used to illustrate the different methods and the proposed algorithms in the thesis. The example is valid for different robots, and implementations with robots with different numbers of degrees-of-freedom are considered in the thesis. Also different sensor configuration setups are considered, both using a wrist-mounted force/torque sensor and a force/torque sensor mounted in the workcell.

The assembly scenario

An assembly graph for the scenario is displayed in Fig. 3.1, i.e., a description of in which order the involved parts should be assembled. The assembly scenario offers a number of smaller sub-assemblies that are considered in the thesis.

The snapfit assembly The upper left part of Fig. 3.1 shows the parts for the snapfit assembly scenario. The dark gray switch should be placed in one of five available slots in the light gray bottom box. Each slot is slightly wider than the switch length, but flexibility in the switch makes it possible to snap it into the correct position.

The red button assembly The parts for this scenario are displayed in the upper right part of Fig. 3.1. First the red button should be in-

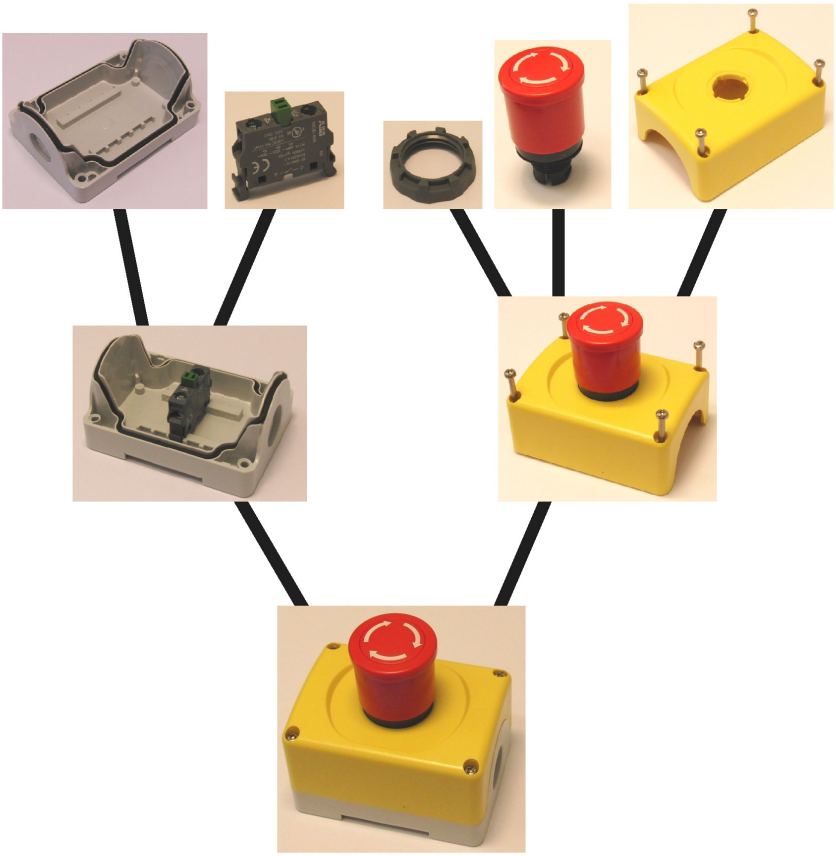

Figure 3.1 Assembly graph for the emergency stop button assembly scenario.

serted 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 complete assembly With the two sub-assemblies performed, i.e., the snapfit and the red button assemblies, the final part 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.

Conclusions

This application case has been one of the main demonstrators in the Rosetta project [ROSETTA, 2012], and it has, e.g., been demonstrated at the Automatica trade fair in Munich 2012. The scenario has been considered in numerous publications, e.g., [Stolt *et al.*, 2011; Björkelund *et al.*, 2011; Stolt *et al.*, 2012a; Stolt *et al.*, 2012b].

4

Assembly framework

Traditional position controlled robotic tasks are specified as trajectories for the robot to follow. This way of specifying the robot motion is not very good when additional sensing is introduced, as it is difficult to relate the sensor measurements to the task specification, i.e., the position trajectory to follow. It is further not easy to reuse the task specification if it only is a set of points to traverse. The task specification in this thesis is based on the constraint-based task specification framework [De Schutter *et al.*, 2007], or simply iTaSC (instantaneous Task Specification using Constraints). It is a general framework that can be used to specify all kinds of robotic tasks. It makes it possible to incorporate different types of sensors, and also take geometric uncertainties into account.

An early framework for specifying force controlled tasks is via hybrid position/force control [Raibert and Craig, 1981]. A framework for specifying end-effector based motion tasks is the operational space formulation [Khatib, 1987], where generalized task specification matrices are used. The task frame formalism [Mason, 1981; Bruyninckx and De Schutter, 1996] is another framework for specifying sensor based tasks.

4.1 Task modeling

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, which was presented in Chapter 3.

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. 4.1. The second object frame, $o2$, is chosen to coincide with the flange frame of the robot.

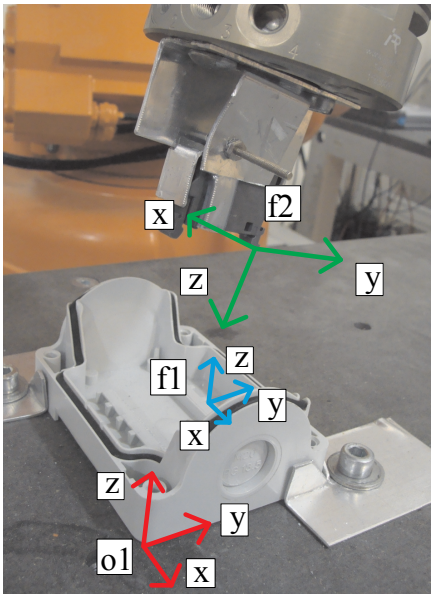

Figure 4.1 Illustration of the coordinate frames in the snapfit scenario. Object frame $o2$ is placed at the flange of the robot, not displayed in this photo.

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 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 $o2$. 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. 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

$$\chi_{fI} = (-) \quad , \quad \chi_{fII} = (x, y, z, \varphi, \theta, \psi) \quad , \quad \chi_{fIII} = (-) \tag{4.1}$$

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 Chapter 6.

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

$$y = f(q, \chi_f) \tag{4.2}$$

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

$$y = \chi_f \tag{4.3}$$

A kinematic chain is closed, which implies a relation between the robot joint coordinates, the feature coordinates, and the uncertainty coordinates. This relation can be expressed as

$$l(q, \chi_f, \chi_u) = 0 \tag{4.4}$$

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

$$\begin{aligned} &T_w^{o1'}(q)T_{o1'}^{o1}(\chi_u)T_{o1}^{f1'}(\chi_{fI})T_{f1'}^{f1}(\chi_u)T_{f1}^{f2}(\chi_{fII})... \\ &T_{f2}^{f2'}(\chi_u)T_{f2'}^{o2}(\chi_{fIII})T_{o2}^{o2'}(\chi_u)T_{o2'}^{w}(q) = I_{4\times4} \end{aligned} \tag{4.5}$$

4.2 Control

The control in this thesis has been made at the velocity level, which means that the purpose is to derive a control law for the robot joint velocities $\dot{q}$, i.e., a velocity reference $\dot{q}_{ref}$ to send to the low-level joint controllers in the robot. In the sequel, the robot is assumed to track the given references and the subscript ref is dropped. The time derivative of (4.2) is

$$\dot{y} = C_q\dot{q} + C_f\dot{\chi}_f \tag{4.6}$$

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 (4.4) have the time derivatives

$$J_q\dot{q} + J_f\dot{\chi}_f + J_u\dot{\chi}_u = 0 \tag{4.7}$$

where $J_q = \partial l/\partial q$, $J_f = \partial l/\partial \chi_f$, and $J_u = \partial l/\partial \chi_u$ are the Jacobians of l with respect to the different coordinates. The feature coordinate time derivatives $\dot{\chi}_f$ can be solved from (4.7), according to

$$\dot{\chi}_f = -J_f^{-1}\left(J_q\dot{q} + J_u\dot{\chi}_u\right) \tag{4.8}$$

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 (4.8) can now be substituted into (4.6), giving

$$A\dot{q} = \dot{y} + B\dot{\chi}_u \tag{4.9}$$

where $A = C_q - C_f J_f^{-1} J_q$ and $B = C_f J_f^{-1} J_u$.

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

$$\dot{y}_d^0 = \dot{y}_d + C \tag{4.10}$$

where $\dot{y}_d$ is a feed-forward 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

$$\dot{q} = A^{-1} \left(\dot{y}_d^0 + B \hat{\dot{\chi}}_u \right) \tag{4.11}$$

where $\hat{\dot{\chi}}_u$ denotes the estimate of $\dot{\chi}_u$. The calculated value of $\dot{q}$ is sent as control signal to the robot.

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 (4.9). When the task is redundant, or over-constrained, the matrix A will not be square, and hence a pseudoinverse must be used. In case of a redundant task the weighted pseudoinverse $A^\dagger$ in (4.12) can be used. The interpretation is that the optimization problem (4.13) is solved, where M is a positive definite weighting matrix.

$$A^\dagger = M^{-1} A^T \left(A M^{-1} A^T \right)^{-1} \tag{4.12}$$

$$\begin{array}{ll} \text{minimize (over } \dot{q}) & \dot{q}^T M \dot{q} \\ \text{subject to} & A\dot{q} = \dot{y}_d^0 + B \hat{\dot{\chi}}_u \end{array} \tag{4.13}$$

Feedback controllers on task level

As mentioned in the previous section, all constraints have to be given at the velocity 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 for the corresponding output. Scalar controllers were used for each component of the output vector y.

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

$$\dot{y}^i = K(y^i_{ref} - y^i) \tag{4.14}$$

where K is the proportional gain and y^i_{ref} is the 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 feed-forward term was used.

Velocity controller The velocity controller used was based on pure feed-forward, according to

$$\dot{y}^i = \dot{y}_{ref} \tag{4.15}$$

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

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

$$\ddot{y}^i_{des} = \frac{1}{M}\left(F^i - F^i_{ref} - D\dot{y}^i_{des}\right) \tag{4.16}$$

where y^i_{des} denotes the desired value of y^i and F^i denotes the force in the direction of y^i. The parameter M is the virtual mass and D the virtual damping of the impedance that the controller acts like. The given acceleration is integrated to give the velocity to use as constraint. The output of the controller is limited in such a way that no wind-up problems occur.

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 control signal is made continuous by adjusting the initial state of the new controller. This ensures that the transfer becomes bumpless [Åström and Wittenmark, 1996].

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 (4.5) 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. The uncertainty coordinates are first assumed to be constant when the value of χ_f is calculated, and then updated, more about this in Chapter 6.

As it is assumed that the only unknown variable of (4.5) is χ_f, the left hand side can be written as $T(\chi_f)$ and the goal is to achieve

$$T(\chi_f) = I_{4\times4} \tag{4.17}$$

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

$$T_{err} = \begin{bmatrix} R_{err} & t_{err} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{4.18}$$

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

$$J_f \Delta\chi_f = \begin{bmatrix} t_{err} \\ a_{err} \end{bmatrix} \tag{4.19}$$

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

$$\chi_f^{i+1} = \chi_f^i + \Delta\chi_f = \chi_f^i + J_f^{-1}(\chi_f^i) \begin{bmatrix} t_{err}^i \\ a_{err}^i \end{bmatrix} \tag{4.20}$$

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 (4.2).

4.3 Software implementation

The assembly framework was implemented using Mathworks' Matlab/Simulink environment. Executable code was generated via the Real-Time Workshop toolbox [Real-Time Workshop, 2012].

An assembly task is coordinated with a finite state machine. Both statecharts using Mathwork's Stateflow [Stateflow, 2012] and sequential function charts using JGrafchart [JGrafchart, 2012] 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 is communicated, i.e, which type of controller to use. Finally, 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 is such that it can be integrated with a standard ABB RAPID program [ABB, 2012]. In this way, initial positioning and picking of parts can be made with the specialized position control architecture in the ABB controller. When external sensing is needed, the execution can be handed over to the external controller, and the execution can be handed back to the ABB controller once the task using external sensing is finished.

5

Snapfit assembly scenario

5.1 Introduction

Assembly tasks can be implemented using position controlled robots if the accuracy of the workcell and the robot, including grippers, is good enough. What is meant with good enough varies, but for small-parts assembly tasks it is usually in the sub millimeter scale. This makes it really difficult, and also costly, to achieve the desired accuracy. An alternative solution is to introduce a force sensor. The measured contact forces can be used to detect different contact situations and in this way compensating for errors in position. It is also possible to relax the position accuracy requirements when a force sensor is available. Uncertain object locations can be handled by implementing the assembly tasks as sequences of search motions, where some of the position uncertainty is resolved by each particular search motion. The measured forces are used to detect when contacts are established, and also to keep contacts once they have been established. The sequencing can be handled by a finite state machine.

This chapter considers an example of a force controlled assembly task, namely the snapfit scenario described in Chapter. 3. The assembly is performed with a sequence of search operations with force measurements triggering transitions. The condition for the assembly being finished is that the switch snaps into place in the bottom box, and a machine learning approach is applied for detecting this event from the

measured force/torque signature.

A survey of the requirements for autonomous robotic assembly is given in [Bruyninckx *et al.*, 2001]. 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 [Jörg *et al.*, 2000], which describes an assembly scenario where sensor fusion of vision and force sensing were used to insert cylinders into a rotating engine. Yet another example from the automotive industry is [Gravel *et al.*, 2008], which describes powertrain assembly. An example 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 [Olsson *et al.*, 2010], where force control was used to avoid sliding movements when drilling.

Machine learning approaches within assembly has previously been used for instance for detection of failures, e.g., in [Cho *et al.*, 2005] and [Hsueh and Yang, 2008] support vector machines were used to detect tool breakage in milling processes. Another example is given in [Rodriguez *et al.*, 2010], which used a single axis force sensor to detect failures in an assembly scenario. A somewhat different approach is 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 is 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.

5.2 Assembly strategy

The modeling of the task is given in Sec. 4.1. Uncertainties in part locations and gripping makes it impossible to use pure position control to accomplish the assembly task. The strategy is therefore to use a sequence of search motions, where each of these motions are designed to resolve some of the uncertainty. The sequencing is modeled by a finite state machine [Gill, 1962].

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 on 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.

For the snapfit scenario it was assumed that the position and orientation of the of the bottom box was not known well enough 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, and the slot was found by two successive search motions (while contact was kept with the bottom). Next, 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 which direction 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. The state machine used for implementing this sequence is displayed in Fig. 5.1.

5.3 Snap detection

The transition condition between states 7 and 8 in the state machine (Fig 5.1) is that the switch is snapped into place. This event can be detected by monitoring the force/torque signals. A machine learning approach was applied to classify wheter a snap had occurred in a given sequence of force/torque data samples.

A total number of 160 executions where the snap occured were recorded. Two different switches were used, and both ends of the switch were used for equally many snaps. To find more robust classifiers, two different robots were used, IRB140 and FRIDA, and also two different rotational velocities during the switch insertion.

Data pretreatment

The force/torque data from all recorded snaps are displayed in Figs. 5.2 (the slow snaps) and 5.3 (the fast snaps). All snaps have been manually

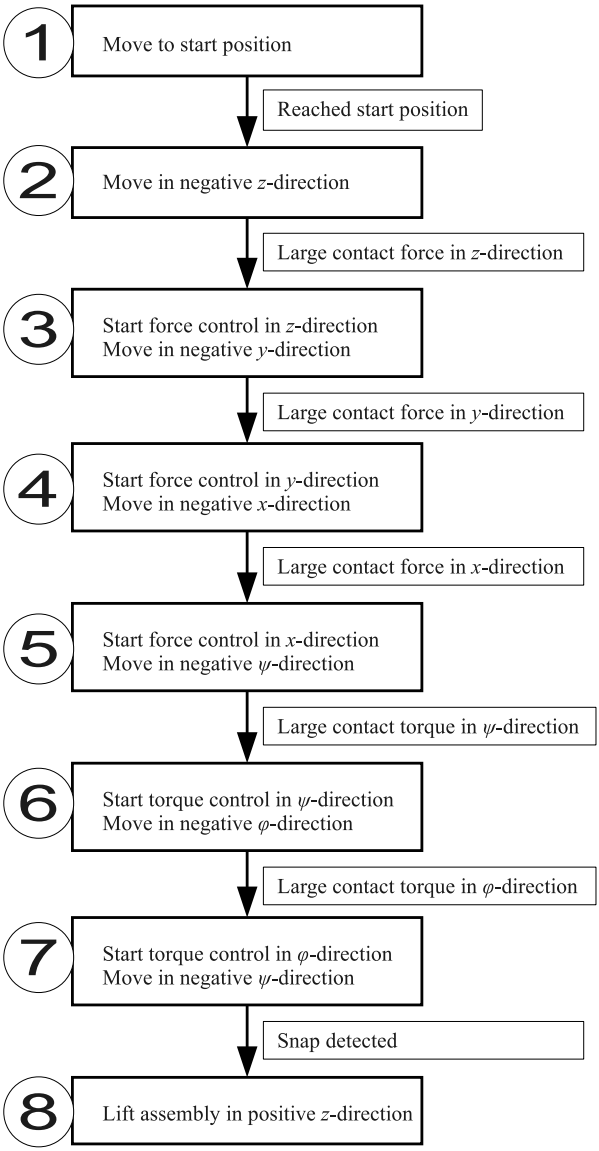

Figure 5.1 State machine describing the assembly sequence.

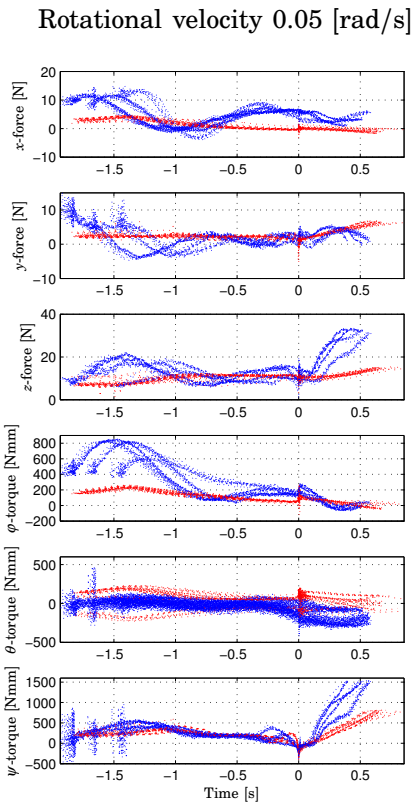

Figure 5.2 Collected data from all sequences where a slow insertion velocity was used. The IRB140 was used for the blue curves and FRIDA for the red curves.

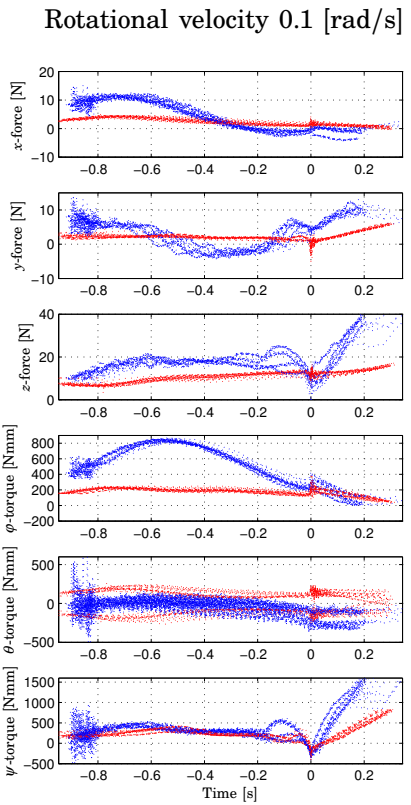

Figure 5.3 Collected data from all sequences where a fast insertion velocity was used. The IRB140 was used for the blue curves and FRIDA for the red curves.

marked, and all sequences have been positioned in the figures such that the snap occurs at $t = 0$ [s]. The snaps recorded with the IRB140 are shown in blue and the ones with FRIDA in red, and it can be seen that these curves have slightly different behaviors. The use of a stiff robot, such as IRB140, gives larger force variations, as forces builds up

quickly when contacts are established. A more compliant robot, such as FRIDA, is more forgiving, i.e., the forces builds up much slower, and therefore gives less variation in the measured force/torque. Further, the time scales in Figs. 5.2 and 5.3 are different, but the shapes of the measured data still look similar. It is possible to see some difference, due to the use of the same controllers. See, e.g., the intial large φ-torque that takes approximately 1 second to reduce in both the diagram with the slow rotational velocity and the diagram with the fast rotational velocity. Finally, the easiest way to recognize the snap with the human eye is probably to look in the ψ-torque plot.

Six different channels in the measured data were available, 3 forces and 3 torques. Only a subset of these channels was considered in the classifiers. A number of n_{pre} samples before the snap and n_{post} samples after the snap were used in each sequence, and the data from the different channels was put after one another in a single vector. The problem was to classify whether such a vector contained a snap or not. Data from different channels was rescaled to get approximately the same magnitude, and the mean value was removed from each sequence, to make the classifier independent of the offset force/torque.

Background data for training Half of the data set was used for training and the other half for validation. The recorded data consisted of all force/torque data from state 7 in the assembly sequence. The transition condition used for leaving the state during data recording was that the ψ-torque was large, and this transition condition can also be used as a backup trigger in cases when the snap detection fails. The recorded data contained a vast amount of background data, i.e., sequences not containing a snap, as every recording only contained one snap. To get a reasonable training data set, a number of N (usually around 20) randomly selected background sequences from each recording together with the actual snap was used for training. Once a particular classifier was trained, it was validated on all sequences in the training data. If any misclassified sequence was found, it was included in the training data set and the training was performed once again. This procedure was iterated until all training data sequences were correctly classified, or that a maximum number of iterations were reached.

The procedure described in the previous paragraph was applied be-

cause it was too computationally expensive to perform training with all available background sequences included, but it was computationally cheap to perform validation on all background sequences. Therefore it was possible to start with a random selection of background sequences, and then add any background sequences that was initially misclassified to the training data set, i.e., sequences were added that were important to have in the training data set to get a robust classifier.

Classifiers considered

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

Least-squares The first classification method tested was based on least-squares. The model used was linear and given by

$$y(x) = w^T x + w_0 = \begin{bmatrix} w^T & w_0 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} = \tilde{w}^T \tilde{x} \tag{5.1}$$

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

All training data can be described by

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{T} = \underbrace{\begin{bmatrix} \tilde{x}_1^T \\ \tilde{x}_2^T \\ \vdots \\ \tilde{x}_n^T \end{bmatrix}}_{\tilde{X}} \tilde{w} \tag{5.2}$$

and by using a sum of squares error function

$$J(\tilde{w}) = \sum_{i=1}^{n} (y_i - \tilde{x}_i^T \tilde{w})^2 = (T - \tilde{X}\tilde{w})^T (T - \tilde{X}\tilde{w}) \tag{5.3}$$

the optimal $\tilde{w}$ is given by

$$\tilde{w} = \left(\tilde{X}^T \tilde{X}\right)^{-1} \tilde{X}^T T \tag{5.4}$$

Classification is now performed by calculating $y(x)$, and the classification boundrary is by default 0, i.e., $y(x) > 0$ means that the data vector is classified as a snap.

Support vector machines The simplest form of support vector machines (SVM) uses the model (5.1), but instead of choosing the parameters based on the least-squares error function, 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

$$\begin{aligned} \underset{\text{over } w,\zeta}{\text{minimize}} \quad & C\sum_{i=1}^{n} \zeta_i + \tfrac{1}{2}\|\tilde{w}\|_2^2 \\ \text{subject to} \quad & t_i y(x_i) \geq 1 - \zeta_i \quad , \quad i = 1,\ldots,n \\ & \zeta_i \geq 0 \quad , \quad i = 1,\ldots,n \end{aligned} \tag{5.5}$$

where ζ_i are slack variables that allow for misclassifications, but the sum of them are punished by the parameter C. The optimization problem (5.5) is convex, and it can therefore be solved easily. This classifier will be called the primal SVM.

By considering the dual formulation of (5.5), it can be shown that the following problem is equivalent to solve

$$\begin{aligned} \underset{\text{over } a}{\text{minimize}} \quad & \sum_{i=1}^{n} a_i - \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j t_i t_j k(x_i, x_j) \\ \text{subject to} \quad & 0 \leq a_i \leq C \quad , \quad i = 1,\ldots,n \\ & \sum_{i=1}^{n} a_i t_i = 0 \end{aligned} \tag{5.6}$$

where the parameters a_i are the Lagrange multipliers corresponding to the inequality constraints $t_i y(x_i) \geq 1 - \zeta_i$ in (5.5). 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 (5.6) was therefore used together with a Gaussian kernel, $k(x_i, x_j) = e^{-(x_i - x_j)^T (x_i - x_j)/l}$. The classification function is now given by

$$y(x) = \sum_{i=1}^{n} a_i t_i k(x, x_i) + b \tag{5.7}$$

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

$$b = \frac{1}{N_S} \sum_{i \in S} \left(t_i - \sum_{j \in S} a_j t_j k(x_i, x_j) \right) \tag{5.8}$$

where S is the set of all support vectors and N_S is the total number of support vectors.

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 used in the snap detection scenario was to find a simple threshold for one of the coordinate axes. All axes were considered, and the one giving the best separation (concerning the importance weights) was used.

Training procedure

In the snapfit example, a false positive, i.e., a detection of a snap by the classifier that actually was not a snap, is much worse than a false negative, i.e., missing to detect a snap. This is due to the fact that there is a backup classifier, based on a simple torque threshold, that can be used if the snap is missed. To take this asymmetry in the problem into account, the cost function (5.9) was used during evaluation of the

different classifiers, where n_{fp} denotes the number of false positives and n_{fn} the number of false negatives.

$$J = 10n_{fp} + n_{fn} \tag{5.9}$$

The cost for false positives is thus 10 times higher than the one for false negatives. In cases where the classifiers did not manage to classify all training data sequences correctly, the final threshold was chosen by minimizing the cost function (5.9).

The first part of the training procedure was to decide which channels of the available data to use, and also how many samples before and after the snap to use. This was done in three steps and evaluated using the least-squares and the boosting classifier.

Varying the channels used First, the number of channels was varied, and the number of samples before and after the snap was set to a high value (30 was used). The result of this experiment on validation data is displayed in Figs. 5.4 and 5.5. All possible combinations of channels to use were tested, and, e.g., the upper left diagrams display the six different possibilities of choosing one channel. Good classification can be achieved with as few as 2 channels, but perfect classification was achieved first when 3 or more channels were used. Therefore, 3 channels was chosen to be used in the final classifier. There was, however, no set of 3 channels that resulted in perfect classification for both least-squares and boosting. A compromise was to choose the channels containing the y- and z-force, and the ψ-torque.

Varying n_{post} The number of samples to use after the snap should be kept to a minimum, to not delay the detection of the snap more than necessary. To find out how many samples after the snap that are needed for good classification, an experiment where this parameter was varied was performed. The number of samples before the snap was large ($n_{pre} = 30$), such that all information before the snap was kept, and the purpose of the experiment was to see how much information after the snap that was needed. The result from the experiment is displayed in Fig. 5.6. The cost is decreased when the number of samples used is increased, which is intuitive as the classifier then has more information available. Taking the result from both tested classifiers into account, it was reasonable to use 6 samples after the snap.

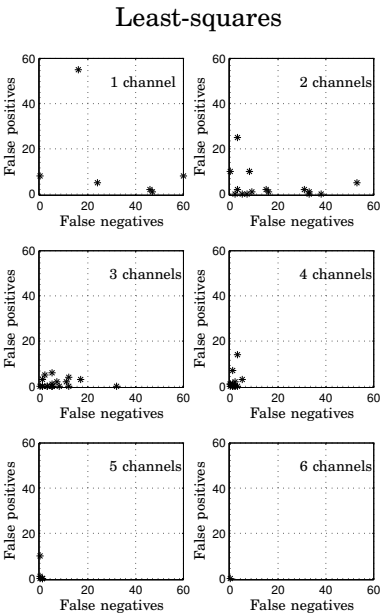

Figure 5.4 Results on validation data when the number of channels used for classification was varied. A least-squares classifier was used.

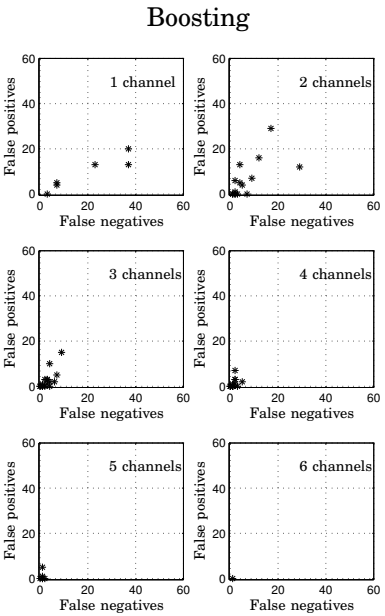

Figure 5.5 Results on validation data when the number of channels used for classification was varied. A boosting classifier was used.

The large jumps in the cost can be explained by the definition of the cost function (5.9). An extra false positive makes the cost increase 10 units, and this is, e.g., what happened at $n_{post} = 5$ in the least-squares result in Fig. 5.6 (the upper diagram).

Varying n_{pre} The third parameter to choose was the number of samples to use before the snap. The trade-off for this parameter consisted in performance versus the complexity of the classifier, more data used gives longer time for training and classification. The result from an experiment where the number of samples used before the snap was varied is displayed in Fig. 5.7, the number of samples after the snap was fixed to $n_{post} = 6$. As the performance is only slightly increased by

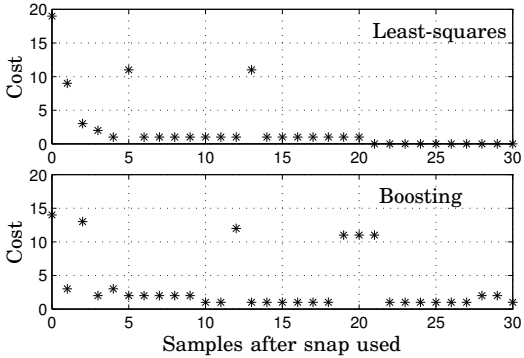

Figure 5.6 Variation of the number of samples after the snap used for classification.

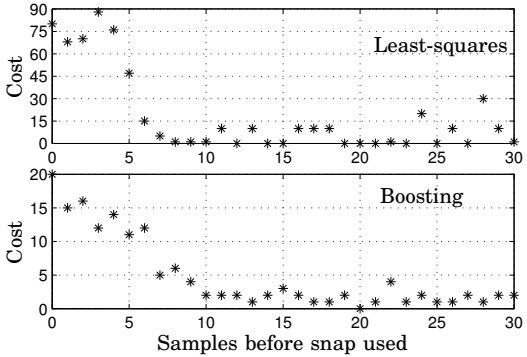

Figure 5.7 Variation of the number of samples before the snap used for classification.

including more samples, it can be seen that 10 samples is a reasonable choice.

Classifier-specific tuning The final stage of the training procedure was to find parameters for the individual classifiers. The least-squares classifier contained no tunable parameters, and neither did the boosting classifier (a fixed number of 50 weak classifiers were used). The SVM classifiers, however, had some parameters.

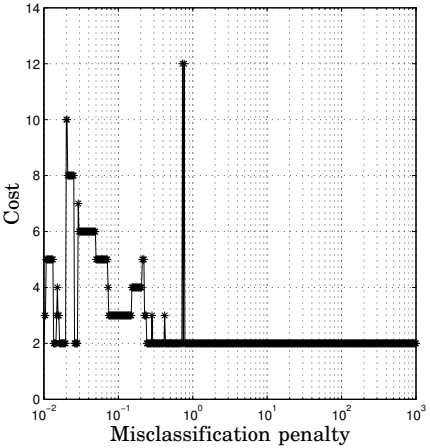

Figure 5.8 Variation of the misclassification penalty in the SVM classifier.

For the primal SVM classifier, the penalty for misclassification, C in (5.5), had to be chosen. The performance on validation data when this parameter was varied is displayed in Fig. 5.8. Clearly, a large misclassification penalty is to prefer. A low penalty makes it affordable to allow for data points to be within the margin and also to be incorrectly classified, for the purpose of increasing the margin. The drawback was, however, that the performance on validation data was decreased. A misclassification penalty of 10 was chosen for the final classifier.

For the SVM classifier using the dual formulation, two parameters had to be chosen, namely the width l of the Gaussian kernel and the misclassification penalty C in (5.6). The result from an experiment where these parameters were varied is displayed in Fig. 5.9. Small kernel widths give classifiers that classify everything as the background, which can be seen as the lower half of Fig. 5.9. Increasing the kernel width gives better performance, at least for large misclassification penalties. The minimum cost was found for a kernel width of approximately 42 and a misclassification penalty of 672.

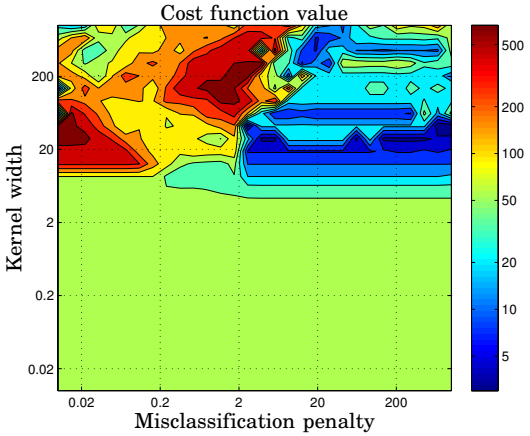

Figure 5.9 Variation of the parameters for the SVM classifier using a Gaussian kernel.

Final comparison

The classification results on training data for all the considered classifiers are displayed in Figs. 5.10-5.13. The last training samples for each classifier correspond to the sequences that initially were misclassified, i.e., those sequences that were added to the training data set during the training procedure (as was described in the 'Background data for training'-section on page 33). Those samples can be seen to all lie relatively close to the decision threshold. A smaller number of background sequences were used for training the SVM classifier using the dual formulation, as the training was very time consuming. Only the boosting and the primal SVM classifier succeed in correctly classifying all training sequences, while both the least-squares and the SVM classifier using the dual formulation make some errors. It can further be seen in all diagrams that the first half of the data[1] gives a larger variation than the second. This outcome was expected, as the first half corresponds to the data from the IRB140 executions, which in Figs. 5.2 and 5.3 was seen to have more variation than the FRIDA executions.

The result on validation data is summarized in Table 5.1. Included

[1]Excluding the last 5-20 samples in the end of each diagram that was added during the training procedure as was described earlier in the paragraph.

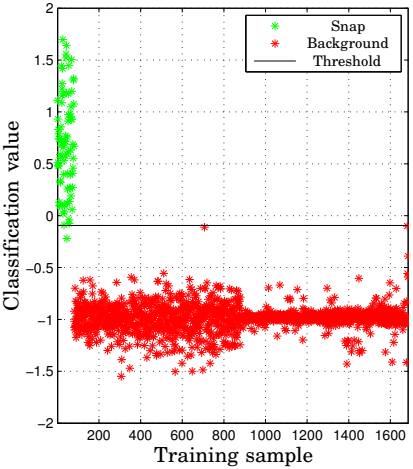

Figure 5.10 Classification on training data for the least-squares classifier.

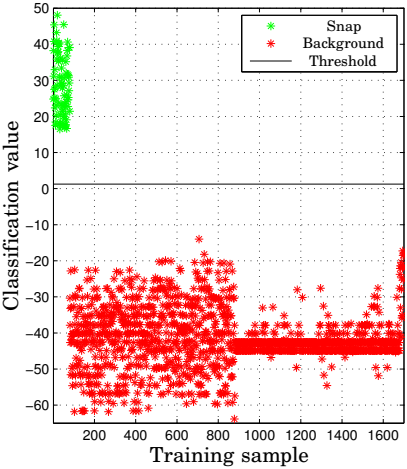

Figure 5.11 Classification on training data for the boosting classifier.

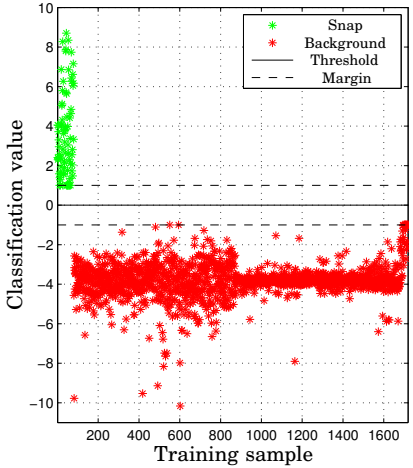

Figure 5.12 Classification on training data for the primal SVM classifier.

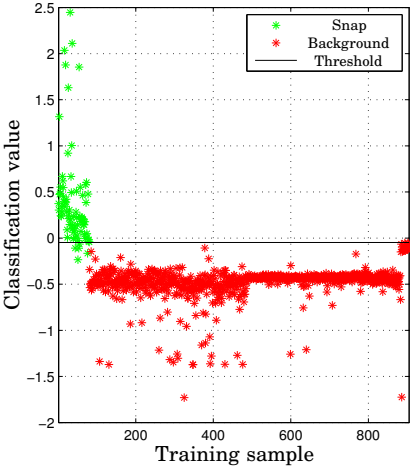

Figure 5.13 Classification on training data for the SVM classifier using the dual formulation with a Gaussian kernel.

Classifier	Threshold	Value	False negatives	False positives
Least-	+10%	0.23	14	0
squares	nominal	-0.095	1	0
	-10%	-0.42	0	34
SVM	+10%	1.89	14	0
(primal)		0.95	7	0
	nominal	0.0	2	0
		-0.95	0	19
	-10%	-1.87	0	153
SVM	+10%	0.37	46	0
(dual)	nominal	-0.048	3	0
	-10%	-0.47	0	17129
Boosting	+10%	12.4	6	0
	nominal	1.25	2	0
	-10%	-9.94	1	9

Table 5.1 Results on validation data. There was a total number of 80 snaps, i.e., the maximum number of false negatives is 80, and the backround data consisted of 31651 sequences, i.e., the mamximum number of false positives is 31651.

is also the performance when the threshold has been increased and decreased with 10 % of the total variation in the training data. This is a measure of the sensitivity of the classifier, a small decrease of the performance when the threshold is changed indicates that the classifier has a good robustness. This measure was, however, unfair to the primal SVM classifier, as the increased and decreased threshold ended up outside the margin. Therefore, also thresholds within the margin has been included.

The least-squares classifier has the best nominal performance, but it is quite sensitive to variations of the threshold. The best classifier is instead the boosting classifier, which only shows a slight decrease in performance when the threshold is changed. The primal SVM classifier

has a similar performance as the boosting classifier, but the dual SVM classifier performs much worse. The use of a Gaussian kernel seems not to be the best alternative, and another choice of kernel function would maybe perform better.

The detection of snaps is quite reliable, concerning the large variation in both training and validation data, with different robots, switches, and assembly speeds. The fact that the least-squares classifier works really well shows that the problem is almost possible to solve using a simple hyperplane classifier. This was an unexpected outcome of the experiments. Using more advanced classifiers, such as boosting, however, gives an increased robustness towards errors in classification.

In a real scenario, the robot user would have to manually mark the snap (or the signature of the event to detect) in a number of sequences. Then the system should perform the entire training procedure automatically. All classifiers considered are possible to implement in a real-time setting, as the calculations required for classifying a data sequence are quite modest.

5.4 Experimental results from an assembly execution

The assembly scenario was implemented using both IRB140 and FRIDA. Exactly the same assembly strategy could be used for both robots, but some of the force control parameters should be modified for good performance, due to different robot properties. The use of a weaker robot, such as FRIDA, makes it easier to accomplish the assembly, as the robot itself is compliant and therefore enables the use of higher search speeds as forces do not build up as quickly as when a stiff robot is used.

Force data from an experimental execution where IRB140 was used is given in Fig. 5.14, together with the corresponding state in the assembly sequence. The first three states shown are linear search motions in z, y and x. The transition conditions were large contact forces in the corresponding search directions, which is easily seen in the force plot, at $t = 1.1$ [s] in z, at $t = 2.0$ [s] in y, and at $t = 2.5$ [s] in x. States 5 and 6 were rotational searches, the transition conditions were large corresponding torques. Notice how the ψ-torque (around $f2$ x-axis) drops around $t = 4.5$ [s]. This was because the switch slid down into the slot in the box. State 7 was the push-down state, and when the snap

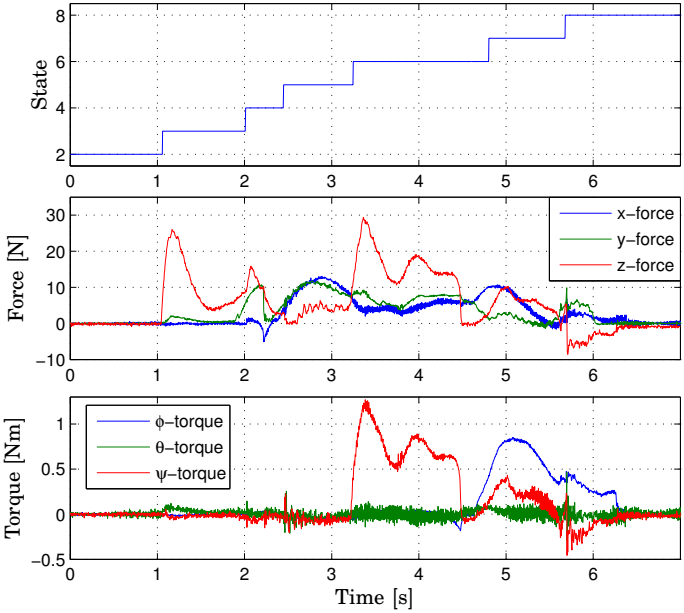

Figure 5.14 Force data from an assembly sequence vs. time. The uppermost diagram shows the state sequence, the middle the forces and the lowermost the torques.

occurred (at $t = 5.7$ [s]) a transition to the last state was made, where the robot lifted the switch and the box to show that it had finished the assembly.

More insight is given in Fig. 5.15, where velocity data from the experiment is shown. Measured data is given when the corresponding coordinates were position or velocity controlled, but the control signal (the desired velocity) is given when the coordinates were force controlled. The search motions are easy to recognize on the non-zero velocities.

The boosting classifier trained in Sec. 5.3 was used for detecting the snap. The resulting classification in each sampling instant is displayed in Fig. 5.16. The snap was detected a $t = 5.68$ [s], as can be seen when the classification value exceeds the threshold. It can further be seen that the background gets approximately the same values as the

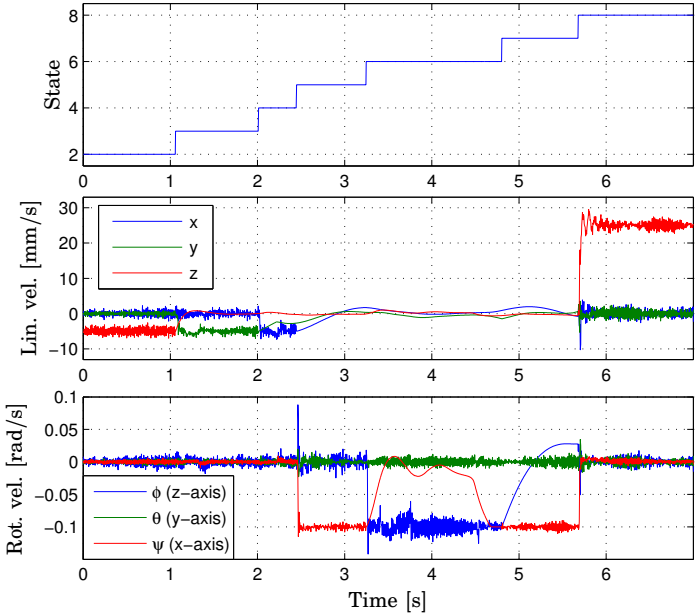

Figure 5.15 Velocity data from an assembly sequence vs. time. The uppermost diagram shows the state sequence, the middle the linear velocities and the lowermost the rotational velocities.

classification on training data in Fig. 5.11, but the snap gets a considerably lower value. The switch used was not one of those used for training, which is a hint for that the classifier may have been somewhat overtrained, i.e., that more than two switches might be needed for training.

5.5 Conclusions

Force control was used to successfully perform the snapfit assembly scenario using two different robots. A machine learning approach was applied for detecting when the switch snaps into place. The classifier with the best performance was a boosting classifier, with simple thresholds along the coordinate axes as weak classifiers.

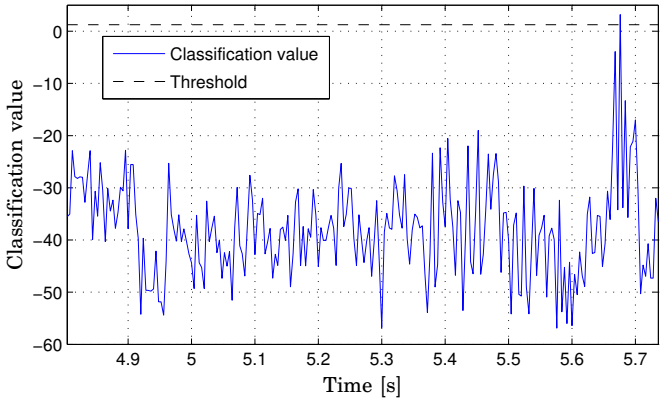

Figure 5.16 Classification results for the boosting classifier during the assembly experiment (when in state 7). The snap was detected when the classification value exceeded the threshold at $t = 5.68$ [s].

6

Uncertainty estimation

6.1 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 system like this has difficulties in handling uncertainties in the task, and fixtures and other task specific solutions are used to avoid problems with them, 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 incorporate 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.

This chapter describes how uncertainties in robotic assembly can be modeled and resolved using force sensing. An example of gripping uncertainty from the snapfit assembly scenario is presented. The uncertainty is resolved using a Kalman filter.

6.2 Modeling

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

The state update and estimation procedure 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 uncertainties makes it possible to achieve a decreased failure rate, and it can also make it possible to decrease the cycle time of the task.

6.3 Example from snapfit assembly

Uncertainties in the task include the exact location of the box and its orientation. They are however resolved using guarded search motions, i.e., the motion is velocity controlled in the search direction and stopped when a contact force is detected. Once contact is made, it is

maintained by using force control, and hence no explicit uncertainty coordinates are used to model this uncertainty. It could be modeled with an uncertainty frame positioned in the true position of the slot. The position of this frame could be estimated when contact had been established in the different directions.

The exact grasp of the switch is also assumed to be uncertain, and the z- and y-distance from $f2$ to $o2$ (Fig. 6.1) is therefore modeled as 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.

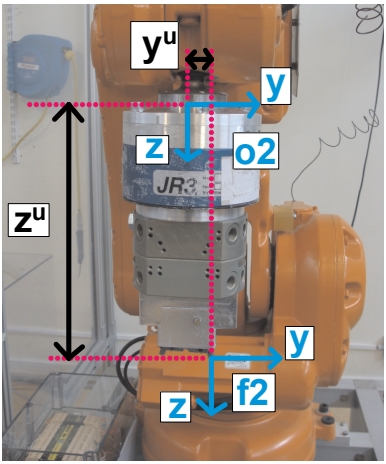

Figure 6.1 Illustration of the uncertainty coordinates y^u and z^u between the two frames $f2$ and $o2$.

An illustration of the uncertainty coordinates y^u and z^u is given in Fig. 6.1. 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 Fig. 6.2) while keeping the switch in contact with the box (this corresponds to state 5 in the assembly sequence, see Sec. 5.2). 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 velocity references in the y and z directions to maintain the forces.

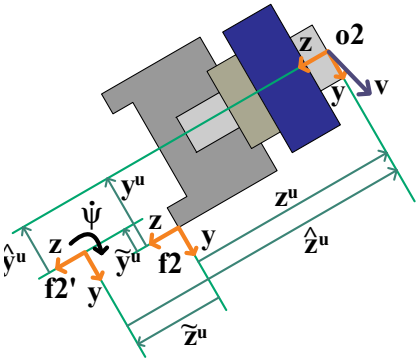

Figure 6.2 Illustration of the uncertainty coordinates y^u and z^u.

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. 6.2, 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

$$v = [\dot{\psi}, 0, 0]^T \times [x_{dist}, y^u, z^u]^T = [0, -\dot{\psi} z^u, \dot{\psi} y^u] \tag{6.1}$$

This assumption only holds 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.

The assumption described in the previous paragraph can be used to set up a dynamical model for y^u and z^u, according to (6.2), 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.

$$\begin{cases} \dot{s} = g(t) \\ m = -\dot{\psi} s + n(t) \end{cases} \tag{6.2}$$

A Kalman filter [Kalman, 1960] can be used to estimate z^u. A discretized model of (6.2) is (6.3), with system matrices defined in (6.4). The noise covariances are assumed to be given by Eq. (6.5).

$$\begin{cases} s(k+1) = As(k) + w(k) \\ m(k) = C(k)s(k) + e(k) \end{cases} \tag{6.3}$$

$$A = I_{2\times 2} \quad , \quad C = \begin{bmatrix} 0 & 0 \\ 0 & -\psi \\ \psi & 0 \end{bmatrix} \tag{6.4}$$

$$\begin{aligned} E\left[w(k)w^T(k)\right] &= Q(k) \\ E\left[w(k)e^T(k)\right] &= 0 \\ E\left[e(k)e^T(k)\right] &= R(k) \end{aligned} \tag{6.5}$$

A Kalman filter for the model (6.3) is given by (6.6)-(6.12).

$$\hat{s}(k|k-1) = A\hat{s}(k-1|k-1) \tag{6.6}$$

$$P(k|k-1) = AP(k-1|k-1)A^T + Q(k) \tag{6.7}$$

$$\tilde{m}(k) = m(k) - C(k)\hat{s}(k|k-1) \tag{6.8}$$

$$S(k) = C(k)P(k|k-1)C^T(k) + R(k) \tag{6.9}$$

$$K(k) = P(k|k-1)C^T(k)S^{-1}(k) \tag{6.10}$$

$$\hat{s}(k|k) = \hat{s}(k|k-1) + K(k)\tilde{m}(k) \tag{6.11}$$

$$P(k|k) = \left(I - K(k)C(k)\right)P(k|k-1) \tag{6.12}$$

6.4 Experimental results

The estimation scheme described in the previous section was implemented in a snapfit assembly like scenario. 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. 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

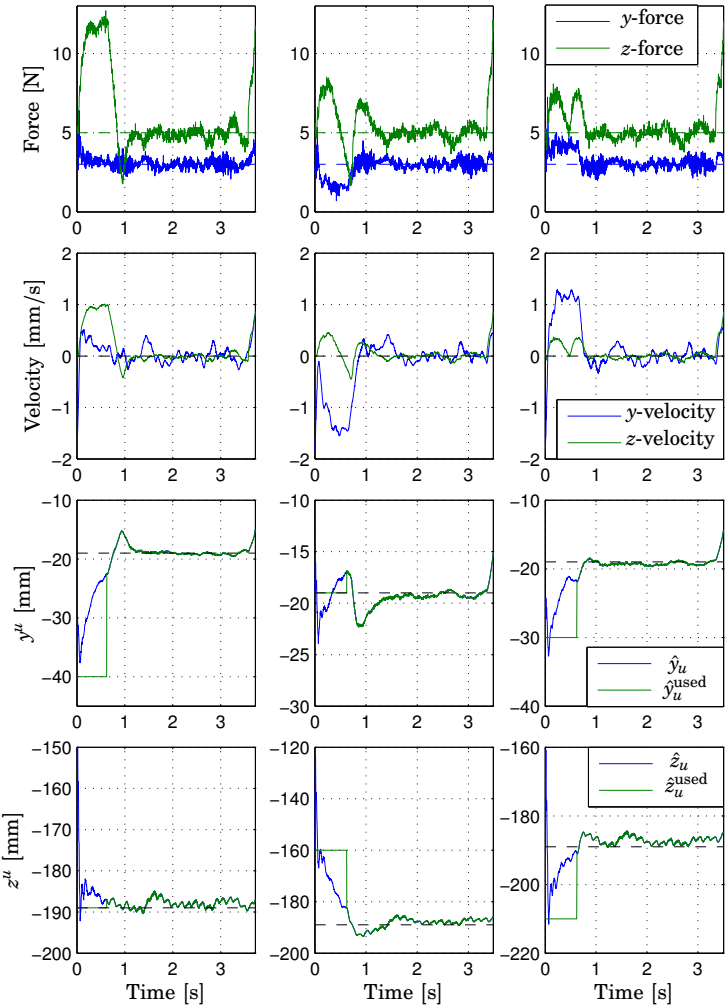

Figure 6.3 Experimental data from the three uncertainty estimation experiments. The uppermost diagrams show the y- and z-forces, measured data with solid lines and references with dashed lines. The middle diagrams show the y- and z-velocities, i.e., the force controller outputs, which should be equal to zero (indicated by black dashed lines). The lowermost 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 until the estimation covariance has settled down.

guesses were used and the result is displayed in Fig. 6.3, where each column of diagrams corresponds to one initial guess.

The state and measurement covariances (6.5) were chosen to be constant and diagonal. The covariance was further chosen to be larger for the measurement noise than for the state noise, i.e., trust the model more than the measurements.

All three experiments started with relatively large force transients. The explanation for this is 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.

6.5 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. Making it completely automatic, i.e., generating an estimator to an uncertainty coordinate specified by the user, is, however, probably not possible. The user will have to also specify the measurement model.

A gripping uncertainty in the snapfit assembly scenario was resolved using a Kalman filter. The method was experimentally implemented on an industrial 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. 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 pre-defined 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 (Chapter 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 pre-

sented in [Love and Book, 1995], which describes how the parameters in an impedance controller can be chosen when using contact model parameters estimated with Recursive Least Squares (RLS). 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 approach was presented in [Kröger *et al.*, 2004], which considers adaptive force controllers within the Task Frame Formalism. 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. Thus, interacting with the environment gives the reaction force, F, given by

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

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

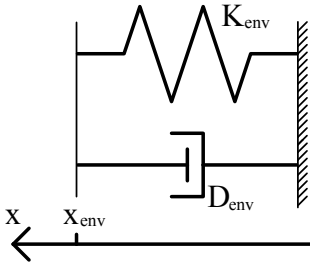

Figure 7.1 Contact model.

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 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. 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

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

The RLS algorithm is given as [Johansson, 1993]

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

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. The initial value of the adaptation gain matrix P has to be chosen, and its magnitude is usually chosen to be large to get a fast convergence to the true values of the estimated parameters.

Each force controlled direction will have nominal parameters, likely not well tuned. These will be used during the estimation of the contact parameters. 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 direction controlled 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 occur. 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 control 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 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

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

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

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

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 (estimated values of contact stiffness and damping should be used)

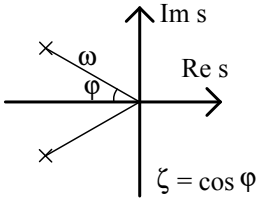

Figure 7.2 Illustration of pole placement design (7.8).

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

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 in the previous subsection can once again be used, with the stiffness $\hat{K}\hat{L}$ and the damping $\hat{D}\hat{L}$, where $\hat{L}$ is the estimated distance between the two contact points, and $\hat{K}$ and $\hat{D}$ 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. An alternative is to instead control the force in each contact. 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 a reference on the forces in each contact to an equivalent force and torque.

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. 4.1, with kinematic chains and the coordinate frames illustrated in Fig. 4.1. The assembly strategy used was the one presented in Sec. 5.2.

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.

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

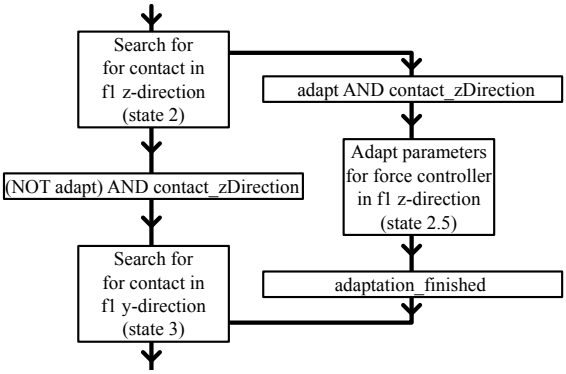

Figure 7.3 A part of the state machine implementing the assembly sequence. The parameter *adapt* decides whether the adaptation phase should be entered or not.

poorly tuned parameters, i.e., a default initial setting, and the adaptive algorithm was initiated. The given force reference 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 $\omega = 5$ [rad/s] and a relative damping $\zeta = 0.8$ was chosen for the controller.

Experimental data from the experiment is shown in Fig. 7.4, the left column shows the measured force and the force reference, and the right column shows the estimated stiffness and damping. In the topmost 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 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, 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

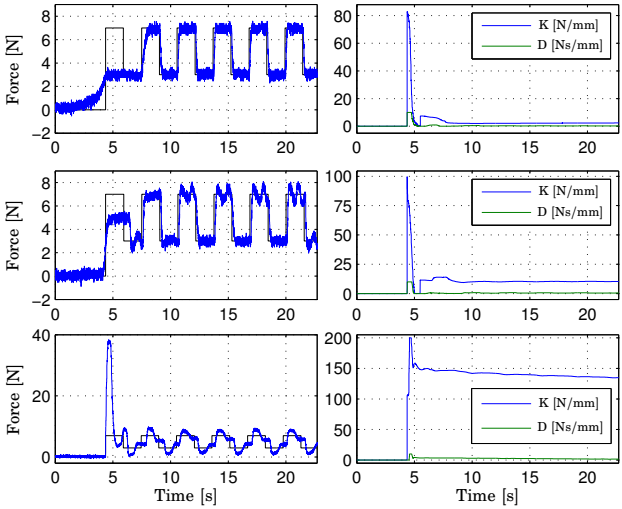

Figure 7.4 Experimental data from an experiment where contact was made with different environments. The top diagrams show contact with soft plastic foam, the middle diagrams contact with a mouse pad, and the bottom diagrams contact with a table surface. The left diagrams show the measured force in blue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

environment was a table surface, 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 starts with a large transient for all materials, which is 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 is displayed in Figs. 7.5-7.8. Force data from the beginning of the sequence is shown in Fig. 7.5. State 2 was the search motion in $f1$ 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) 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 parameter around $t = 14$ [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

[1]This state corresponds to the adaptation state. It was defined in Fig. 7.3.

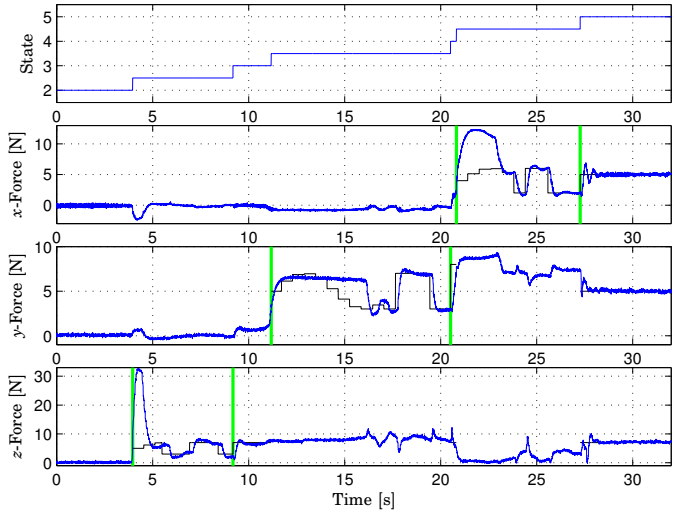

Figure 7.5 Experimental data from the beginning of the assembly sequence. The top diagram shows the state sequence, with state numbers defined in Sec. 5.2, and the decimal states defined according to Fig. 7.3. The remaining diagrams show the measured force (blue) and the force reference (black) for the 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.

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 $t = 4$ [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 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

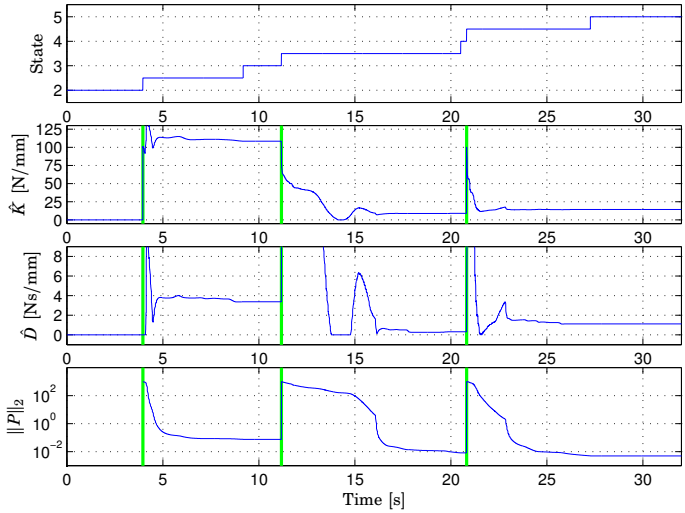

Figure 7.6 Experimental data from the beginning of the assembly sequence. The top diagram shows the state sequence, the second the stiffness parameter estimate, the third the damping parameter estimate, and the last diagram the norm of the P-matrix. The beginning of each adaptation phase is marked with 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.

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 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.

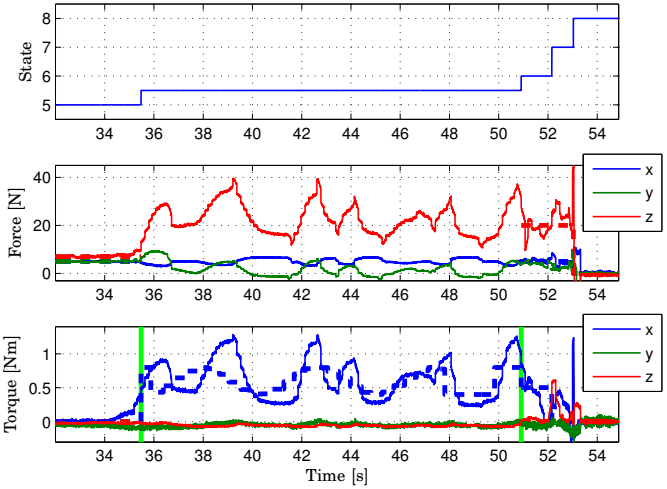

Figure 7.7 Experimental data from the final part of the assembly sequence. The top diagram shows the state sequence. The second diagram shows the measured force (solid lines) and the force reference (dashed lines) for the coordinate directions in frame $f1$. The third diagram shows the measured torque (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.

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 (Sec. 7.2) was implemented in the assembly sequence. The only relevant state in the sequence was state 6, and experimental data from this state is shown in 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

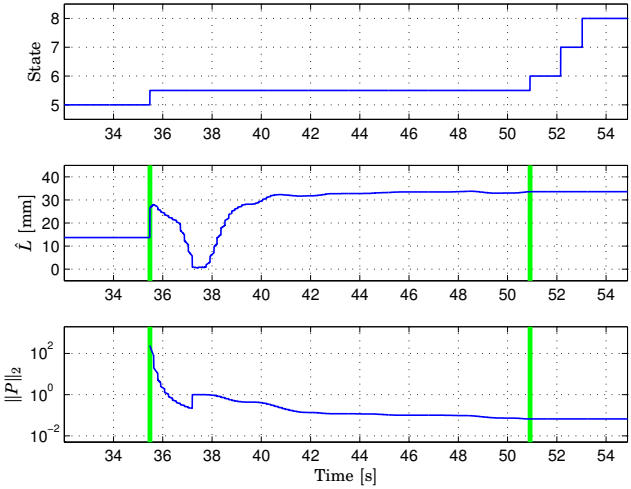

Figure 7.8 Experimental data from the final part of the assembly sequence. The top diagram shows the state sequence, the second diagram the distance parameter estimate (between the contact points), and the last diagram the norm of the P-matrix. The adaptation phase is marked with green lines.

caused by the switch sliding down into the slot, i.e., contact was lost for the second point. This can also be seen in the torque diagram, as the torque approaches 0. 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 is satisfactory, both for soft and stiff contacts, and it can be used to free the user from the tedious work of tuning the force controllers manually. Some performance degradation for stiff contacts is 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

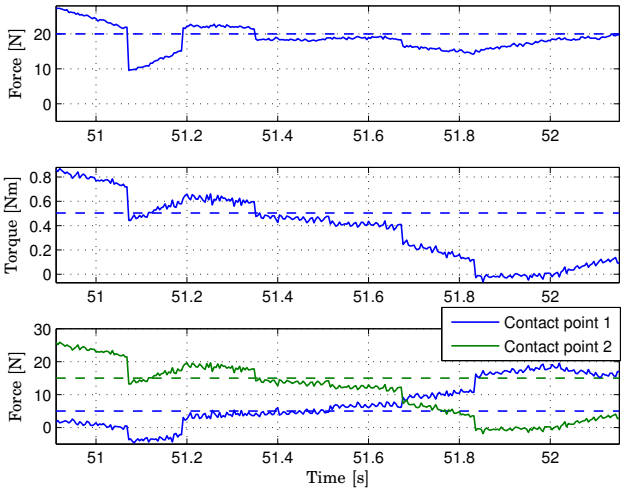

Figure 7.9 Experimental data from state 6 in the assembly sequence. The top diagram shows the z-force, the middle diagram the x-torque, and the bottom diagram the estimated equivalent forces acting on the two ends of the switch. Measured force/torque are shown with solid lines and references with dashed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

considers the limitations of the robot system, also the robot dynamics has 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 difficulty 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 is currently implemented as separate states in the controlling state machine. A dedicated excitation signal is used to assure that input data to the estimation algorithm is sufficiently exciting. A further development could be to run the adaptation algorithm in each assembly operation without an excitation signal.

The contact locations have been estimated during the assembly sequence, but this information has so far not been used. One way to use it is to decrease the search times, by starting the search motions closer to the identified contact locations, after a number of successful assembly operations have been performed.

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.

To the best of the authors knowledge, a similar approach has not been previously presented within assembly. Adaptive force control with comparable 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 in an industrial robot system and used in an assembly task.

8

Singularity-free orientation representation based on quaternions

8.1 Introduction

This thesis focuces on creating a framework for sensor-guided assembly that should be possible to use for non-expert robot operators. In Chapters 4 and 5 the snapfit assembly scenario was presented. The implementation was based upon the iTaSC framework, where Euler angles commonly are used as a representation for orientation. A main reason for using this representation is that it is intuitive to work with and that it offers a minimal representation for orientations, but unfortunately it has problems with representation singularitites [Siciliano and Khatib, 2008].

One way to avoid the problems with an Euler angle representation, but keeping the intuitive orientation description, is to use an internal singularity-free representation. Two such representations are quaternions and rotation matrices. The former representation is the one considered in this thesis, due to the fact that a quaternion needs 4 parameters only, whereas a rotation matrix needs 9 (not all unique) ones. Another possibility is to switch between different Euler angle

representations that have the representation singularity in different orientations, as described in [Singla *et al.*, 2005]. A drawback with a switching solution is that it would be cumbersome to handle all possible Euler angle parametrizations, if the user should have the possibility to choose from all of them. The mapping of constraints between the switching representations might also be a problem. An advantage with the quaternion representation is that it always gives orthogonal rotation axes, which is not the case for an Euler angle representation. If the task requires constraints on non-orthogonal rotation axes, the remedy is to use more than one kinematic chain.

A general method to handle non-minimal representations, such as quaternions, within iTaSC is suggested in [De Schutter *et al.*, 2007]. Both this approach and the one presented in this thesis use the fact that there exists a minimal representation on the velocity level. In fact, they can be proven to be equal. This thesis further presents how the quaternion representation is integrated into the iTaSC methodology, making it possible for the user to specify tasks without having to worry about orientation representation singularities. The approach is experimentally verified in an implementation of a force-controlled assembly task. The task considered is a subassembly of an emergency stop button; namely the initial part of the red button assembly described in Chapter 3. The red button should be inserted into the yellow case, i.e., a peg-in-hole assembly. The button is assumed to be rotationally symmetric, which introduces a redundant degree of freedom in the task. This redundancy is exploited using the iTaSC framework.

Quaternions have previously been used in many contexts, such as robotics, computer graphics, and attitude control of aircrafts and spacecrafts. A primary reason for working with quaternions is that they are a singularity-free orientation representation. An underwater vehicle-manipulator system was controlled in a singularity-free manner using the unit quaternion in [Antonelli and Chiaverini, 1998], and the quaternion was also used in [Caccavale and Siciliano, 2001] to avoid representation singularities in control of a redundant manipulator on a spacecraft. In [Wen and Kreutz-Delgado, 1991] the unit quaternion was used as a singularity-free orientation representation to derive a large family of globally stable control laws for the attitude control problem. Another example where the quaternion is used to avoid singularities is [Xian *et al.*, 2004], where it was used to implement a task-space

tracking control scheme.

Quaternions have also been used in connection with SLAM, e.g., in [Kyrki, 2008] they are used to represent similarity transforms. Further, a general framework on how to handle redundancy is presented in [Rocco and Zanchettin, 2010].

8.2 Preliminaries

Orientation representation

Orientation in the kinematic chains in iTaSC is usually represented by Euler angles, i.e., three consecutive rotations around given coordinate axes. The reason for choosing Euler angles is that they are intuitive to work with and easy to specify in a kinematic chain. Furthermore, they offer a convenient way of parametrizing an orientation, and also make it possible to control all three angles separately.

There are, however, several problems with an Euler angle representation. The first one is that the parameterization is not unique, e.g., $(\frac{\pi}{2}, \frac{\pi}{2}, 0)$ and $(-\frac{\pi}{2}, -\frac{\pi}{2}, \pi)$ are two examples of ZYZ-Euler angles that represent the same orientation. This results in problems when the inverse kinematics problem is considered, i.e., when calculating the Euler angles for a given orientation. Another problem is the inherent representation singularity, which occurs when two rotation axes are aligned. This results in the Jacobian of the kinematic chain losing rank. The iTaSC motion solver uses an inverse of this Jacobian, and a representation singularity is therefore highly inconvenient.

Quaternions

Quaternions [Hamilton, 1840] are an extension of the complex number system. A quaternion Q consists of a scalar part, $Q_s \in \mathbb{R}$, and a vector part, $\bar{Q}_v \in \mathbb{R}^3$, according to

$$Q = \begin{pmatrix} Q_s, & \bar{Q}_v \end{pmatrix} \tag{8.1}$$

More on the properties of quaternions can, e.g., be found in [Siciliano and Khatib, 2008]. Unit quaternions are a suitable choice as a representation for rotations. The rotation around an axis $\bar{v}$, $|\bar{v}| = 1$, with the

angle θ is then given by the quaternion

$$Q = \left(\cos(\theta/2), \quad \sin(\theta/2)\,\bar{v} \right) \tag{8.2}$$

The use of quaternions for representing rotations does not exhibit the problems of the Euler angle representation. The drawback is, however, the non-minimality of the quaternion representation, in which case four parameters are needed together with the normalization constraint, $\|Q\| = 1$. The intuitivity of the Euler angles is also lost.

8.3 Quaternion representation

Kinematics

To be able to incorporate quaternions in a kinematic chain in the iTaSC framework, a new rotation transformation has to be introduced. It is a general 3D rotation, and its current value is described by a quaternion. One difference from other types of transformations previously used within iTaSC is that this has three degrees of freedom, and the corresponding feature coordinate is a 4D-vector. A kinematic chain can contain several of these quaternion transformations, but only one can be part of the feature coordinates. The others might be used to introduce uncertain and constant reorientations.

When formulating the iTaSC motion specification, all constraints are transformed to velocity constraints. Force and position constraints are handled by the use of controllers that output a desired velocity to achieve the constraints. The velocity of an ordinary feature coordinate is simply its time derivative, but for a quaternion transformation this is not the desired way to describe a velocity constraint, as the derivative of the quaternion parameters are even less intuitive than the quaternion itself. Therefore a geometric approach is adopted, and the velocity considered is the angular velocity in the local coordinate system described by the quaternion. This means that the angular velocities around the coordinate axes described by the quaternion are considered as feature coordinates and not the quaternion parameters. The quaternion is used to keep track of the current orientation. Using

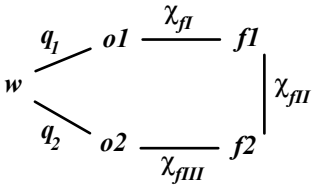

Figure 8.1 The inverse kinematics problem is solved by considering the position loop constraint that is defined by the kinematic chain, w denoting the world coordinate frame, q_1 and q_2 robot joint coordinates, $o1$ and $o2$ object frames, $f1$ and $f2$ feature frames, and $\chi_f = (\chi_{fI}, \chi_{fII}, \chi_{fIII})$ the feature coordinates.

the angular velocities as feature coordinates makes it easy to calculate the part of the Jacobian of the kinematic chain belonging to the quaternion transformation. The Jacobian is given in (8.3), where x_i, y_i, and z_i are the rotation axes, i.e., the coordinate axes in the coordinate system described by the quaternion. The vector $\bar{t}$ represents the vector from the rotation point to the endpoint of the kinematic chain.

$$J_Q = \begin{bmatrix} x_i \times \bar{t} & y_i \times \bar{t} & z_i \times \bar{t} \\ x_i & y_i & z_i \end{bmatrix} \tag{8.3}$$

Inverse kinematics, i.e., the problem of finding the feature coordinates when the rest of the kinematic chain is known, can be solved in a similar way as the model update proposed in [De Schutter *et al.*, 2007]. Consider the position loop constraint (8.4) represented as a product of homogenous transformation matrices[1], see also the illustration in Fig. 8.1.

$$\begin{aligned} T_{w\to o1}(q_1)T_{o1\to f1}(\chi_{fI})T_{f1\to f2}(\chi_{fII})\ldots \\ T_{f2\to o2}(\chi_{fIII})T_{o2\to w}(q_2) = I_{4\times 4} \end{aligned} \tag{8.4}$$

When there is a quaternion Q in the kinematic chain, the feature coordinates can be written $\chi_f = (Q, \bar{\chi}_f)$, where $\bar{\chi}_f$ contains all feature coordinates except for Q. An equivalent formulation of (8.4) is (8.5), where the fact that q_1 and q_2 are known and constant has been used.

$$T_1T_2(\bar{\chi}_f)T_3(Q)T_4(\bar{\chi}_f)T_5 = I_{4\times 4} \tag{8.5}$$

[1]The uncertainty coordinates have been omitted here, but the inclusion of them are straightforward as they are considered to be known and constant in these calculations.

This position loop constraint can be solved for $\bar{\chi}_f$ and Q in an iterative fashion. By first assuming that $\bar{\chi}_f$ is known, it is possible to calculate Q such that the orientation part of (8.5) is fulfilled, i.e., only the rotation matrix part of the homogenous transformations is considered. To then solve (8.5) for $\bar{\chi}_f$ one can for instance make a linear approximation. Let us denote the left-hand side of (8.5) for $T(\bar{\chi}_f)$ and the current estimate of $\bar{\chi}_f$ for $\hat{\bar{\chi}}_f$, then (8.6) holds, where R_{err} is the orientation error represented by a rotation matrix and t_{err} the translation error represented by a Cartesian vector.

$$T(\hat{\bar{\chi}}_f) = T_{err} I_{4\times4} \quad , \quad T_{err} = \begin{bmatrix} R_{err} & t_{err} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{8.6}$$

A linear approximation to describe $\Delta\bar{\chi}_f = \bar{\chi}_f - \hat{\bar{\chi}}_f$ is (8.7), where a_{err} is an axis/angle representation of R_{err} and $J_{\bar{\chi}_f}$ the Jacobian for $T(\bar{\chi}_f)$.

$$J_{\bar{\chi}_f} \Delta\bar{\chi}_f = \begin{bmatrix} t_{err} \\ a_{err} \end{bmatrix} \tag{8.7}$$

Normally there are 6 feature coordinates, but as $\bar{\chi}_f$ does not contain the quaternion only 3 feature coordinates remain. This means that a least-squares solution can be used to solve (8.7) for $\Delta\bar{\chi}_f$ and update $\hat{\bar{\chi}}_f$.

Iteration of the described procedure is performed until the error, i.e., the right-hand side of (8.7), is small enough. As the inverse kinematics is calculated continuously during operation and the coordinate values in the previous sample are used as starting values in the next sample, usually only one or a few iterations are needed. The only exception is in the start-up of the program, when the initial guess might be far off.

Euler angle references

The value of the quaternion transformation is possible to specify in any format, and it is possible to give a desired orientation in Euler angles. This is no problem as the transformation this way is unique, i.e., all possible Euler angle coordinates for a particular orientation result in the same quaternion. The same holds for velocity and torque

references. In the velocity control case the desired Euler angle velocity is transformed to a desired angular velocity using the Jacobian, relating the Euler angle time derivatives to the angular velocity. When a torque constraint is active the controller output is the desired velocity, which is transformed using the Jacobian, in the same way as the velocity control case.

Hybrid control

Controlling a quaternion in a kinematic chain to a desired value is fairly straightforward. If the current orientation is denoted Q_{cur} and the desired orientation Q_{des}, then the orientation error is given by $Q_{err} = Q_{cur}^{-1} * Q_{des}$, where $*$ denotes quaternion multiplication. By exploiting the fact that the error describes a rotation, its rotation axis $\bar{v}_{err}$ and its rotation angle θ_{err} can be calculated from the parametrization (8.2). The orientation error can now be eliminated by applying the desired angular velocity (8.8), where K is a gain factor.

$$\bar{\omega}_{des} = K\theta_{err}\bar{v}_{err} \tag{8.8}$$

It is a bit more difficult to apply hybrid control, e.g., when it is desired to control the orientation around one axis and torque around another. A solution to this problem is to continuously update the quaternion reference, by integrating the velocity references given by the torque controller. The position (orientation) controller is constrained to only apply velocity corrections around the axes that are position controlled, i.e., the desired velocity from the controller is projected onto the axes that are position controlled. Updating the reference for the part of the orientation that is position controlled requires a complete orientation reference, or a change relative to the current orientation. Only giving the Euler angles for the position controlled directions is not enough, as the complete orientation description can not be uniquely calculated from this information.

The integration of the quaternion reference is made by applying a rotation with constant angular velocity during one sample period, i.e., multiplying with the quaternion (8.9), where $\bar{\omega}$ is the angular velocity and h the sample period.

$$Q_{int} = (\ \cos(|\bar{\omega}|h/2),\quad \sin(|\bar{\omega}|h/2)\bar{\omega}/|\bar{\omega}|\ ) \tag{8.9}$$

Redundancy

A task may not need all available degrees of freedom, and such redundancy should be exploited. If the orientation that is described by a quaternion is part of the redundancy it can be handled by not specifying the velocity around the redundant axis. When the quaternion is position controlled the calculated desired velocity should be projected onto the rotation axes that are part of the task. This means that any desired quaternion can be specified, but that only errors projected into the degrees of freedom being part of the task will be eliminated.

Introducing quaternions do not alter the way iTaSC handles the redundancy, as the quaternion on the velocity level is completely described by three angular velocities.

8.4 Assembly scenario

The assembly scenario used to illustrate the quaternion approach is the first part of the red button assemnly (Chapter 3), i.e., inserting the red button into the yellow case. The button is assumed to be rotationally symmetric, although this is not exactly true. Making this assumption, however, makes the task redundant, as the rotation around the symmetry axis does not matter. If the button is grasped in such a way that the symmetry axis coincides with the last joint axis of the robot, the redundancy is trivial and only results in the position of the last robot joint being unconstrained. The gripper is designed in such a way that the redundant degree of freedom is not trivial. A wrist-mounted 6 degrees-of-freedom force/torque sensor is used to perform the assembly.

Kinematic chain

One kinematic chain is used in the assembly task and the object and feature frames related to it are shown in Fig. 8.2.

- Object frame $o1$ is attached to the box. It is related to the world coordinate frame by a fix transformation.
- Feature frame $f1$ has its origin in the center of the hole on top of the yellow case. The orientation is the same as $o1$.

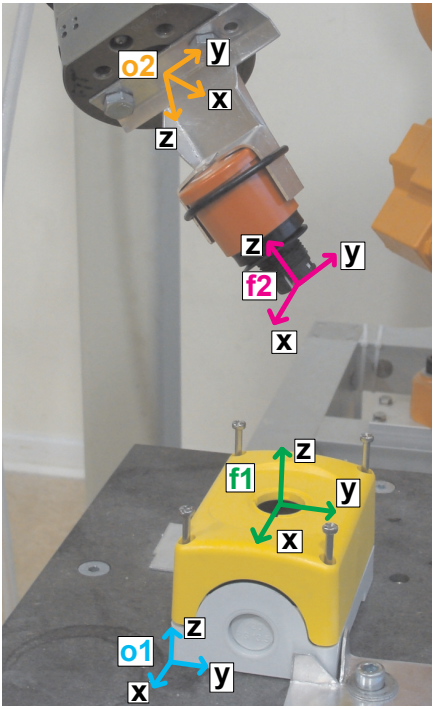

Figure 8.2 Illustration of the different coordinate frames used in the assembly task.

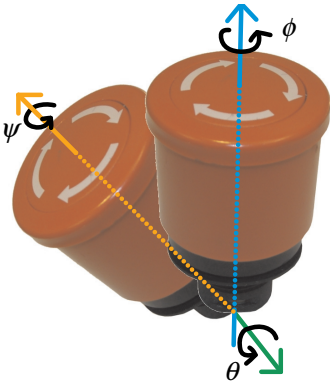

Figure 8.3 Illustration of the Euler angle representation (ZYZ) of the orientation of the button. ϕ is a rotation around the z-axis, θ is then a rotation around the y-axis in the new coordinate system defined by the first rotation. Finally ψ is a rotation around the new z-axis.

- Feature frame $f2$ is attached to the endpoint of the button and its orientation is illustrated in Fig. 8.2.

- Object frame $o2$ has its origin on the base of the gripper and the same orientation as the robot flange frame. It is related to $f2$ by a fix transformation.

The feature coordinates χ_f are divided into three groups depending on which frames they relate to, according to

$$\chi_{fI} = (-) \qquad o1 \rightarrow f1$$
$$\chi_{fII} = (x, y, z, Q) \qquad f1 \rightarrow f2$$
$$\chi_{fIII} = (-) \qquad f2 \rightarrow o2$$

The first three feature coordinates, (x, y, z), are translations along the coordinate axes of $f1$. Then it is intuitive to describe the orientation of the red button with ZYZ-Euler angles, illustrated in Fig. 8.3. First a rotation around the z-axis is made, and then a rotation around the y-axis of the new coordinate system. Finally, a rotation around the new z-axis is introduced. The last axis is the symmetry axis of the button and this rotation thus corresponds to the redundancy in the task. This Euler angle representation would certainly cause trouble if it would have been used for feedback control. The reason is that it has a representation singularity close to the target position, because in this position the second angle coordinate is zero and the first and the third rotation axes coincide. Instead, the three Euler angles are represented by a quaternion, Q, internally.

Outputs are chosen as Eq. (8.10), where the two last outputs are specified on the velocity level and correspond to the quaternion (ω_x and ω_y denote the angular velocities around the x- and the y-axis, respectively, in the coordinate frame described by the quaternion, i.e., frame $f2$). The actual values of y_4 and y_5 are not defined, but the torque corresponding to these outputs is the torque around the corresponding rotation axes (x- and y-axis, respectively). The last angular velocity, ω_z, is not chosen as output as it will not be constrained in any way, due to the assumption of rotational symmetry.

$$y_1 = x\ , \quad y_2 = y\ , \quad y_3 = z\ , \quad \dot{y}_4 = \omega_x\ , \quad \dot{y}_5 = \omega_y \tag{8.10}$$

Uncertainties in the task include the exact location and orientation of the box, and the grasp of the button. They were, however, resolved using guarded search motions, i.e., the motion was velocity controlled in the search direction and stopped when a contact force was detected. Once contact was made, it was maintained by using force control, and hence no explicit uncertainty coordinates were used to model this uncertainty.

Assembly strategy

The assembly strategy is designed in such a way that the uncertainties are resolved during execution of the task. The position of the yellow case is assumed not to be known well enough for hitting the hole with the button in the upright position. But the uncertainty is small enough for an approach with a tilted button to hit the hole (Fig. 8.2). Once the hole is found, force control is used to find the center of it. Then the button is reoriented towards what is assumed to be the upright position while using force control to press downwards, such that the button gradually slides down into the hole. Torque control is then used to find the correct orientation. This strategy is modeled with a state machine, where each state has the following actions:

1. Goto start position
2. Search for contact in z-direction (output y_3)
3. Force control to center of hole
4. Reorient to the approximate upright position
5. Control torques to zero
6. (Release button and) move robot away

Position or force measurements are used to trigger transitions between subsequent states.

8.5 Experimental results

Force data from an experimental execution is given in Fig. 8.4, together with the corresponding state in the assembly sequence. The first state shown, state 2, is the search in the z-direction. The transition condition to the next state is that a large z-force is detected, and this happens at $t = 2.2$ [s]. State 3 is a search for the middle of the hole, which is made by controlling the x- and y-forces to zero while keeping a positive force in the z-direction; the reference is set to 10 [N]. The next step is then a position control of the orientation to the assumed upright position of the button, while pressing in the z-direction such that the button slides down completely into the hole. In state number 5 the torques

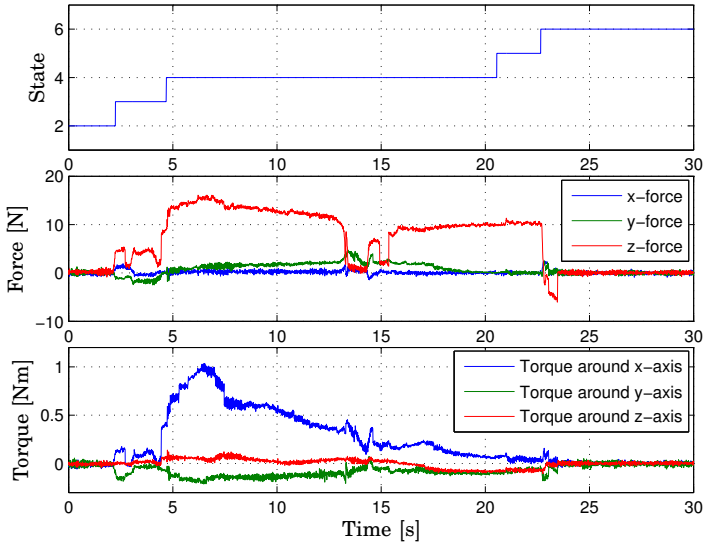

Figure 8.4 Force data from an assembly sequence vs. time. The uppermost diagram shows the state sequence, the middle the forces (along the first three feature coordinate directions, i.e, they are given in frame $f1$) and the lowermost the torques (around the coordinate axes defined by the quaternion, i.e., they are given in frame $f2$).

around the x- and y-axes are controlled to zero, such that the button is completely pushed down into the hole. The last state is that the robot is moved away in the positive z-direction.

The feature coordinates with Euler angles in the kinematic chain have been calculated for the experimental execution shown in Fig. 8.4, and these angles are shown in Fig. 8.5. The initial position chosen was $\phi = -90°$, $\theta = 36°$ and $\psi = 90°$, where the ψ-angle corresponds to the redundant degree of freedom. As the θ-coordinate was decreased, corresponding to the button being moved towards the upright position, the current pose was getting closer and closer to the singular configuration. When it came close, the ϕ- and the ψ-angles immediately changed with about 180°, and the reason it stopped there is probably that the singularity was only closely passed by. If this kinematic chain would have been used for control it would be hard to predict what would hap-

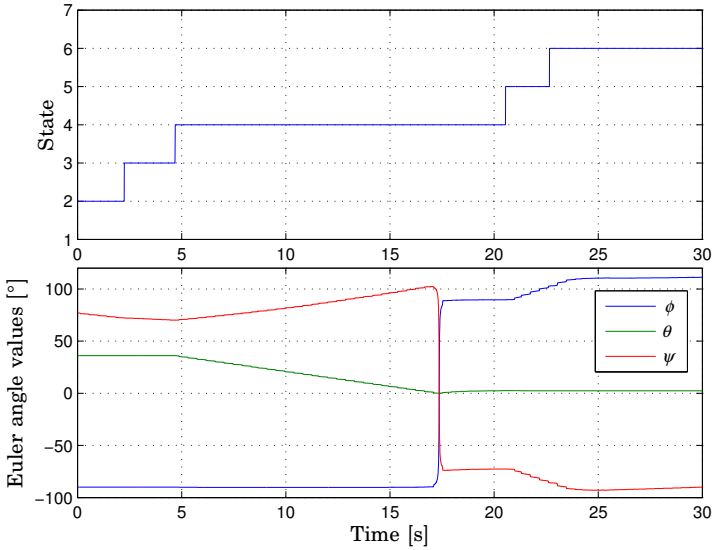

Figure 8.5 Calculated Euler angles from the assembly sequence. ϕ is the first rotation around the z-axis, θ the rotation around the y-axis, and ψ the final rotation around the z-axis. Problems occur when the singular position is entered at $t = 17.3$ [s].

pen close to the singularity. Furthermore, if the program would have survived the singularity, problems might have occurred because the ϕ-coordinate had drifted away.

The redundancy in the task was handled by choosing the weighting matrix M (in Eq. (4.12)) as (8.11), i.e., such that it was desired to rather move the wrist of the robot (joints 4, 5 and 6) than the three first joints.

$$M = \begin{pmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{8.11}$$

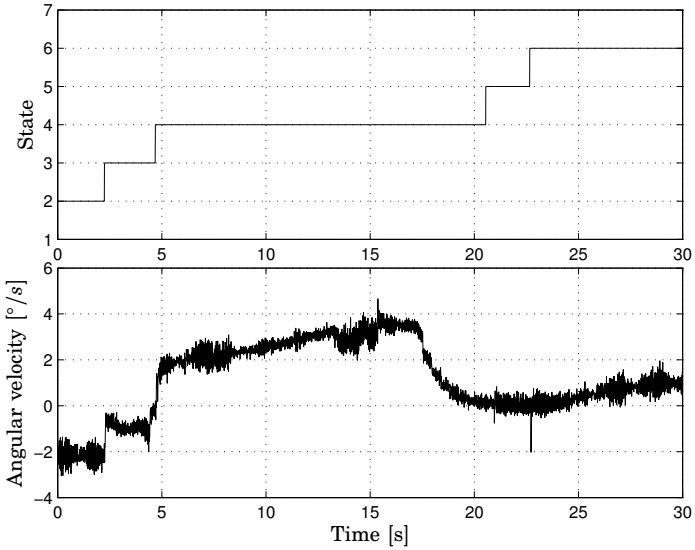

Figure 8.6 Measured angular velocity in the redundant direction.

The resulting measured redundant angular velocity from the experimental execution is shown in Fig. 8.6. Quite large rotations can be seen, indicating that the redundancy was exploited, especially around $t = 16$ [s] when the button is closing in on the upright position and slides down into the hole.

8.6 Discussion

The described choice of Euler angles in the assembly task had a representation singularity close to the target position, and would therefore cause trouble in the execution of the task. The previous solution to this problem was to be careful and choose a safe orientation representation, e.g., as was done in the snapfit assembly scenario described in Chapter 5. An Euler XYZ representation would for instance be fine for the red button assembly scenario in this chapter. However, using the assembly framework and the quaternion representation described relieves the user from doing these considerations, and the user only has

to come up with a suitable representation for the task. In this way it is a step towards making it easier for unexperienced robot programmers to accomplish assembly tasks.

A drawback of the approach with an internal singularity-free representation is that it is not possible to use feedback from the individual Euler angles. An example is when one Euler angle direction is torque controlled and the other two position controlled, as then a reference change for the position controlled angles might be impossible to translate to the internal representation because of the Euler angle ambiguity. These situations are rare, and most scenarios relevant in assembly are not subject to this problem. In case they do occur, the remedy is to specify the complete orientation.

The current experimental implementation of the proposed quaternion representation is not completely automatic, i.e., it is not possible for a user to make the modeling and design the state machine describing the task with Euler angles and get an implementation based on the quaternion representation. The current procedure requires some manual steps, which in a real application have to be made automatic. This is, however, a shortcoming of the implementation, not of the proposed method.

The redundancy resolution should be chosen such that some criterion is optimized, e.g., the energy spent during the task execution could be minimized. The weighting matrix M in the assembly task has been chosen quite arbitrarily, where the only objective was to show the concept of relative weighting. Further efforts should be spent to find a meaningful criterion (4.13).

The button was assumed to be rotationally symmetric, which is not true in reality. There are some edges that might get stuck during the insertion in the hole. This sometimes happened during execution of the assembly. Some extra oscillations are then induced, but otherwise the assembly sequence still works fine.

The current assembly speed is not that fast, but very little effort has been spent on optimizing it. This is, however, something that would be important in an industrial application. If it is assumed that the same assembly is performed several times, learning approaches can be used to generate feed-forward data that can speed up the assembly. Learning can also be used to make changes in the assembly sequence, if this is appropriate.

8.7 Conclusions

A method to introduce a singularity-free orientation representation based on quaternions within the iTaSC framework has been described. The proposed method makes it possible to model tasks with Euler angles, and execute them such that the inherent representation singularities cause no problems. The method has further on been implemented on an industrial robot system and experimentally verified in a force-controlled assembly task. The chosen task contained a redundant degree of freedom that was exploited using the iTaSC framework.

9

Force controlled assembly without a force sensor

9.1 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. To be able to do this one has to rely on that the accuracy of the robot, of the involved parts, and of the workcell are good enough. Usually task specific fixtures and toolings are 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 motions 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 uncertain-

ties, 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 formations, 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.

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 are different. In the case of pure position control one has to assume that the position accuracy is very good 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 feedback control strategies, which may be hard to tune. The two last strategies also require a sensor, which may be expensive. The reusability and the increased robustness to uncertainties, however, are incitements to use the strategy based on continuous sensing.

This chapter will propose a method of how to perform force sensing without a force sensor, by instead estimating the forces from the joint position control errors. The method is experimentally verified in two real-world small-part assembly tasks using a redundant robotic manipulator, see Fig. 2.2.

9.2 Estimating external forces

The simplest and most straightforward way of estimating the external force acting on the end-effector is to use a force sensor. One alternative is to use a wrist-mounted sensor, and by assuming that the end-effector is rigidly connected to the sensor, the external force acting on the robot can be calculated. Another option is to use torque sensors on each joint of the robot, e.g., as done in the DLR lightweight arm [Albu-Schäffer *et al.*, 2007]. If the individual joint torques are collected into the vector τ, the end-effector force F can be calculated from $\tau = J^T F$, where J is

the Jacobian of the robot.

If there is no force sensor available, one alternative is to use the motor torque in each joint. The problem with this approach is that it requires the measured torques to be compensated for disturbance forces—e.g., gravity and friction—before the torque signals can be used. Further problems arise if there are gears in the robot, since a high gear ratio will deteriorate the signal to noise ratio, as applied forces on the arm side are scaled down on the motor side. This will make it hard to achieve an accurate estimate. If it is possible to extract the joint torques, the same approach as with individual torque sensors in the joints can be utilized. 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.

A third 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 is used. In [Hacksel and Salcudean, 1994] and [Alcocer *et al.*, 2003] the force is estimated by considering how position estimation errors behave 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 is 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].

Methods for force estimation are also available in commercial robot systems, e.g., both Toshiba [Toshiba, 2012] and ABB [ABB, 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.

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 then one will probably run into problems with friction, which is hard to model in a good way. Even with all parameters known, for a manipulator with 6 or 7 joints the dynamic equations get very large and complicated. They might therefore be hard to implement in a real-time controller.

Yet another approach 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, they will act as virtual springs, and the deviation of each joint angle from its reference will correspond to a joint torque. By using the same approach as with individual joint torque sensors, it is possible to calculate the external force. 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 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

$$\tau = J^T F + e \tag{9.1}$$

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 $\hat{F} = (JR_e^{-1}J^T)^{-1}JR_e^{-1}\tau$, 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 (9.1), then the minimum variance unbiased estimate of F is

$$\hat{F} = (JR_e^{-1}J^T + R_F^{-1})^{-1}(JR_e^{-1}\tau + R_F^{-1}\bar{F}) \tag{9.2}$$

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.

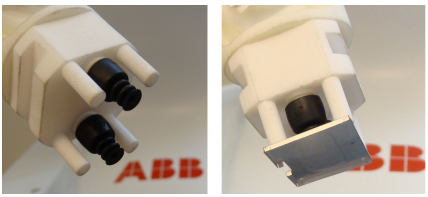

Figure 9.1 The vacuum gripper used in the first assembly scenario. Left: Unloaded gripper. Right: Shield can in the gripper.

9.3 Assembly scenarios

The main assembly scenario considered is a subassembly of a mobile phone. A 'shield can' (metal lid) should be assembled onto a printed circuit board (PCB). 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.

A second scenario used to test the approach is the screwing of the nut in the emergency stop button assembly, see Chapter 3. This particular assembly is made with both of FRIDA's arms.

Robot tooling

To make it possible to perform the mobile phone assembly, special tooling has been produced. A fixture has been designed for keeping the PCB in position. A suction tool is used to grasp the shield can, see Fig. 9.1. The maneuverability in contact is good in the vertical direction (the $f2$ z-direction in Fig. 9.2), but worse orthogonal to this direction ($f2$ x- and y-direction in Fig. 9.2), since the shield can may slide.

A 6 degrees-of-freedom ATI Mini40 force/torque sensor has been placed beneath the PCB fixture to make it possible to both perform the shield can assembly using force sensor feedback and make a comparison with it when the external forces are estimated from the joint errors only.

The nut screwing task uses a three finger gripper to hold the button on one of the arms and a two finger gripper to hold the nut with the

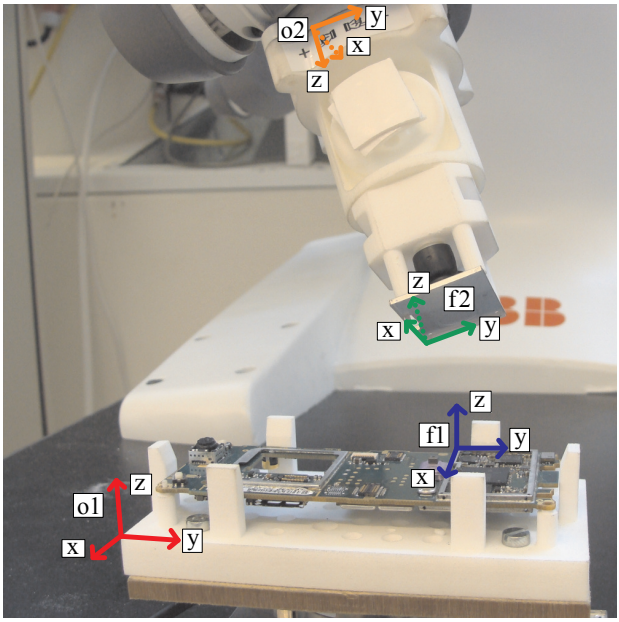

Figure 9.2 Illustration of the frames used in the shield can assembly task.

other arm.

9.4 Task modeling

The tasks were modeled with the iTaSC framework. A thorough explanation of this is given in Chapter 4 and [De Schutter *et al.*, 2007].

Shield can assembly task

One kinematic chain is used in the assembly task and it is illustrated in Fig. 9.2. A schematic description of the kinematic chain is given in Fig. 9.3.

- Object frame $o1$ is attached to the fixture holding the PCB. It is related to the world coordinate frame by a constant transformation.

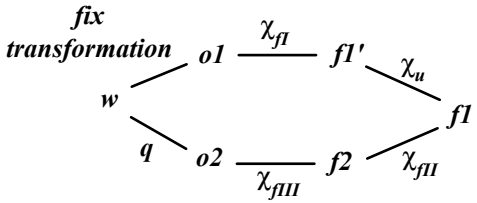

Figure 9.3 Schematic illustration of the kinematic chain used in the shield can assembly task.

- 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.

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

$$\chi_{fI} = (-) \qquad o1 \rightarrow f1$$
$$\chi_{fII} = (x, y, z, \varphi, \theta, \psi) \qquad f1 \rightarrow f2$$
$$\chi_{fIII} = (-) \qquad f2 \rightarrow o2$$

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

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

Uncertainties Uncertainties in the task include the exact location and orientation of the fixture holding the PCB, and also how the shield can has been grasped. The first one can be modeled by introducing an uncertainty frame $f1'$. This frame represents the modeled position of

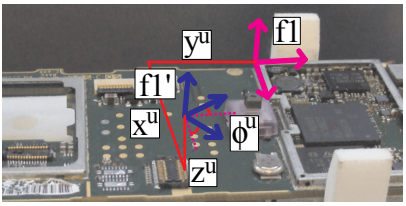

Figure 9.4 Illustration of the uncertainties in the shield can assembly task. The magnitude of the uncertainty is scaled up to make room for the frames and the labels.

the socket corner on the PCB, while $f1$ gives the real position. It is assumed that the fixture is mounted such that the PCB is placed in the horizontal plane, but the exact location and orientation in this plane is uncertain. This is modeled by introducing three uncertain translations and one uncertain reorientation angle (around the $f1$ z-axis), see Fig. 9.4. Similarly, in the grasp the orientation around the z-axis and the translations along the x- and y-axes in frame $f2$ are uncertain.

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

Assembly strategy The assembly strategy is designed such that the uncertainties are resolved in a robust way. A suitable strategy is to first find a corner of the socket with the shield can in a tilted position, see e.g. Fig. 9.2, by executing a sequence of guarded search motions, i.e., the search motions are stopped once the corresponding contact force is sensed. Once the corner is found, rotate the shield can to what is estimated to be the correct orientation and press it onto the socket. When a certain force and torque are applied the shield can can be considered to be mounted correctly.

By inspecting the PCB a suitable corner to try to find is the one where frame $f1$ is placed, see Fig. 9.4. The area in front of this corner is almost free of small edges that can lead to problems during the assembly. It is 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 is found is given in Fig. 9.5.

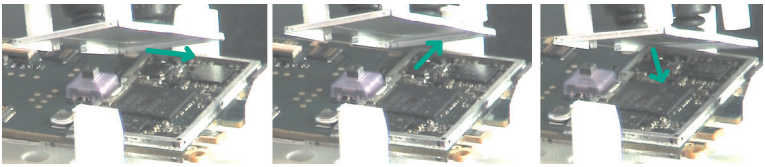

Figure 9.5 Snapshots from the shield can assembly sequence to illustrate how the corner is found. The arrows indicate in which direction the shield can is in contact. In the leftmost photo the robot is in state 5 and has sensed contact in the y-direction, in the middle photo the robot is in state 6 and has sensed contact in the x-direction, and in the rightmost photo the robot has found the corner.

Nut screw scenario

Two kinematic chains are used for the nut screw scenario, see illustration in Fig. 9.6. The first one (chain a) is used to specify the relative motion between the arms, and the second (chain b) to specify 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. 9.7 (here the superscript has been dropped). The world coordinate frame, w, is placed

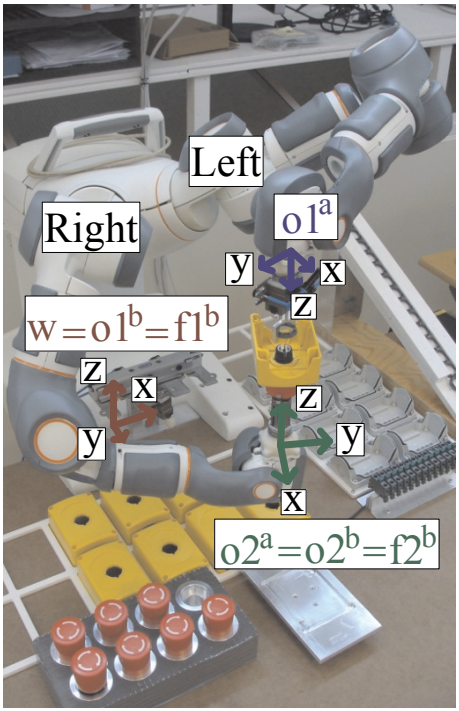

Figure 9.6 Illustration of the two kinematic chains used in the nut screw assembly scenario. The trays for all the different parts in the whole emergency stop button scenario can be seen in the background.

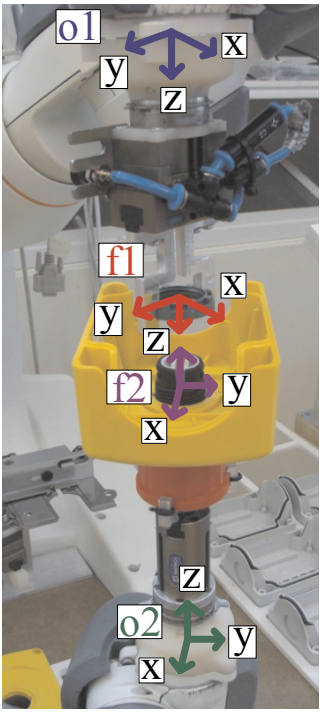

Figure 9.7 A detailed view of the first kinematic chain (chain a) in the nut screw assembly scenario.

at the base of the robot and the chain begins with the left FRIDA arm. Object frame $o1$ is placed on the flange of the robot arm, and feature frame $f1$ in the middle of the gripped nut. Feature frame $f2$ is placed at the bottom of the yellow case, centered in the button. The second object frame, $o2$, is placed at the flange of the right robot arm. The chain is closed by going back to w through the right arm.

The feature coordinates used are 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 describe

the relative distance between $f1$ and $f2$, e.g., a positive z-translation results in the arms moving apart (at least for the configuration shown in Fig. 9.7).

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

Assembly strategy A sketch of the state machine coordinating the assembly task is displayed in Fig. 9.8. The red button with the yellow case on it is placed in a vertical position to start with. The second arm, with the nut in the gripper, approaches the first and tries to put the nut on the button (as in Fig. 9.6). This is made as a search in $f1^a$ z-direction. The search is ended when a contact force is detected, and if the distance between the arms is small enough, the nut was put on the button, otherwise the hole was missed. In the latter case, the robot moves back and modifies the initial position slightly and tries again, and this continues until the nut is on the button.

The gripper for the nut can grasp the nut in two ways. The first one is used for putting the nut on the button, and the second for screwing. The nut therefore has to be released and regripped, and this may cause a movement of the nut. The remedy is therefore to push on the nut to make sure it is all the way down on the button. The next step is to go down to the nut, grip it, and start screwing. The robot can screw one revolution, and then has to release the nut and go back to the start position again.

It is known that it takes 2.5 to 3.5 revolutions to tighten the nut, depending on where on the thread the screwing is started. Detecting that the nut is tightened can not be done through the estimated torque around the screwing axis, due to too large disturbances. What usually happens is that the grip of the nut is lost when it becomes tightened, and then the screwing arm pushes on the arm holding the button, resulting in a large side force which can be detected. If this happens during the third or the fourth revolution of screwing, it is assumed

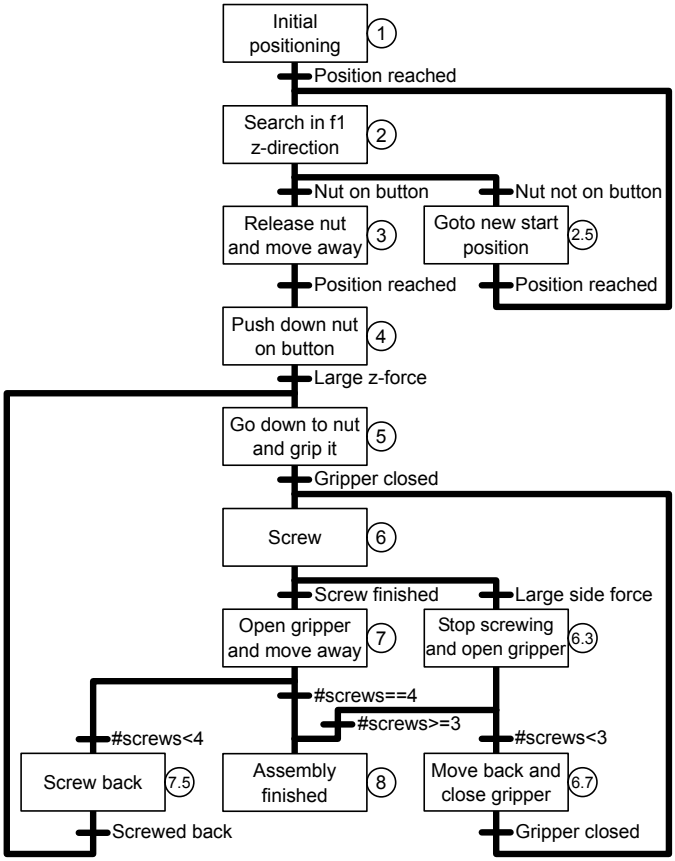

Figure 9.8 The state machine used for the nut screw assembly scenario.

that the reason was that the nut was tightened. If no large side forces occur within four revolutions, the button has most likely slid in the grip, which means that it has been tightened.

During screwing, there is a risk that the nut is gripped in a non-centric way, which may cause a large side force when the screwing is started, although the nut is not tightened. If such a force is detected during the first or the second revolution, the screwing is stopped. The

nut is then released, the robot arms are slightly moved apart, and then the nut is gripped again, and the screwing continued. This set of actions usually leads to a better grip.

The arm holding the button is 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. Another source of errors are the position errors introduced by the detuning of the joint controllers.

No force sensing is available in the arms, and the assembly operation therefore has to rely on the estimated forces.

9.5 Experimental results

Force estimaton

The force estimation is affected by how the detuning of the joint controllers has 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. 9.9, where the force sensor has been 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 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 $0.03K_i$ as integral gain, where K_i is the integral gain in the nominal joint controllers.

A large disturbance acting on the force estimates 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. (9.1). The effect of gravity was assumed to vary slowly, such that the integral part in the joint controllers could

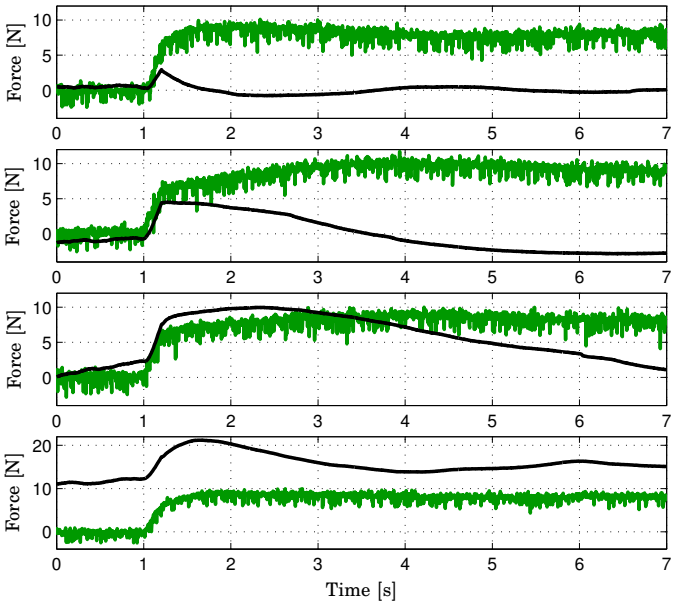

Figure 9.9 Estimated force (black) and measured force (green) in one direction for some different values of the integral gain in the joint controllers. The nominal controllers have the integral gain K_i. The topmost diagram has integral gain K_i, the second diagram $0.1K_i$, the third diagram $0.01K_i$, and the lowermost no integral action at all.

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. 9.11.

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 5 experiments were performed for each arm configuration, and the results can be seen in Fig 9.10. For each joint the mean value of the experiments was later used for force estimation.

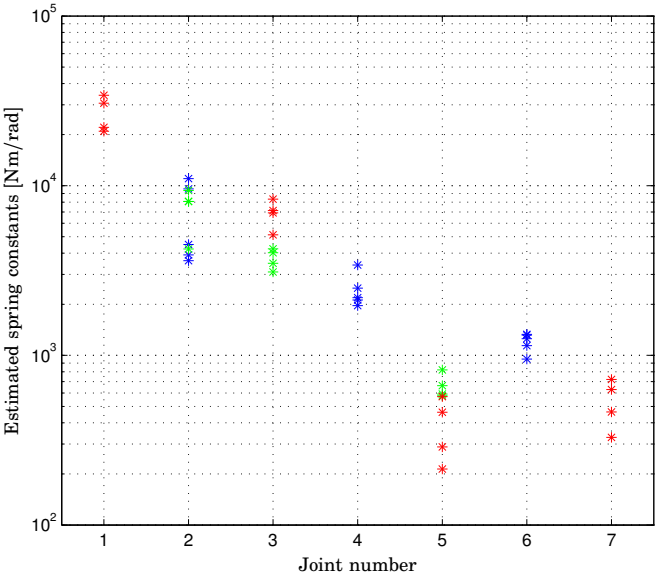

Figure 9.10 Experimentally determined spring constants for the different joints. The joints are numbered from shoulder to wrist. The different colors denote values obtained from different arm configurations.

Shield can assembly scenario

Torques in the assembly sequence were measured in frame $f2$. The major part of the assembly sequence is therefore performed with only a point contact, which means that the torques should be zero around this point. Modeling errors will of course contribute to some torques, but they should be small. This insight can be used as prior information, i.e., it can be used to choose $\bar{F}$ and R_F. No bias force is expected, which gives $\bar{F} = 0$. The variance R_F is chosen to be large for the forces and small for the torques. The estimate that utilizes this prior information is compared to an estimate without this information in connection with Figs. 9.11 and 9.12.

Assembly scenario with force sensor The assembly sequence described in Sec. 9.4 was implemented using a force sensor. The results were similar to those presented in Chapter 5. Major differences were

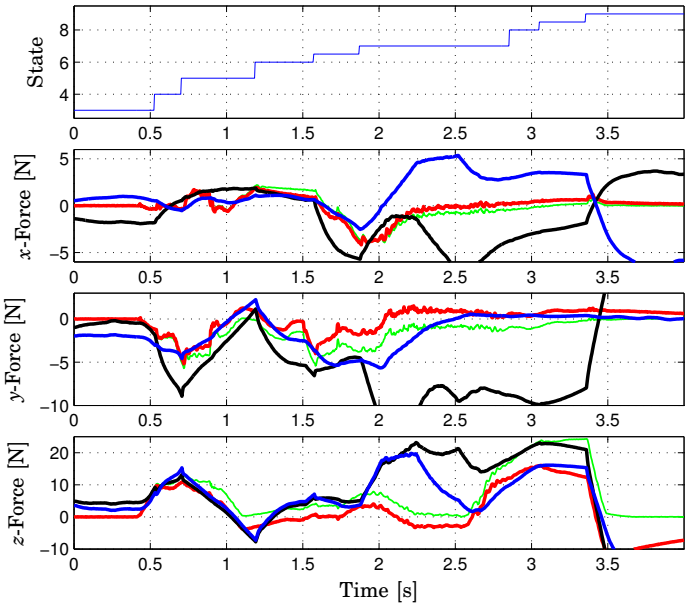

Figure 9.11 Measured and estimated forces from the the assembly sequence without force sensor together with the state sequence. Forces estimated with a priori information about the low torques are shown in blue, and forces estimated without a priori information in black. For reference, raw force data from the force sensor is shown in green, and high-pass filtered force data in red. The data from the force sensor has not been used for control in this execution.

that the involved parts were smaller here, and also the magnitude of the measured forces. The sliding search motions caused large disturbance forces that made it hard for the force controllers to keep contacts in other directions. This made it difficult to increase the overall assembly speed. The small force magnitudes also required care when choosing force thresholds for the state transitions in the controlling state machine.

Assembly scenario without force sensor The assembly strategy 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. Instead, once a search motion made contact the position was

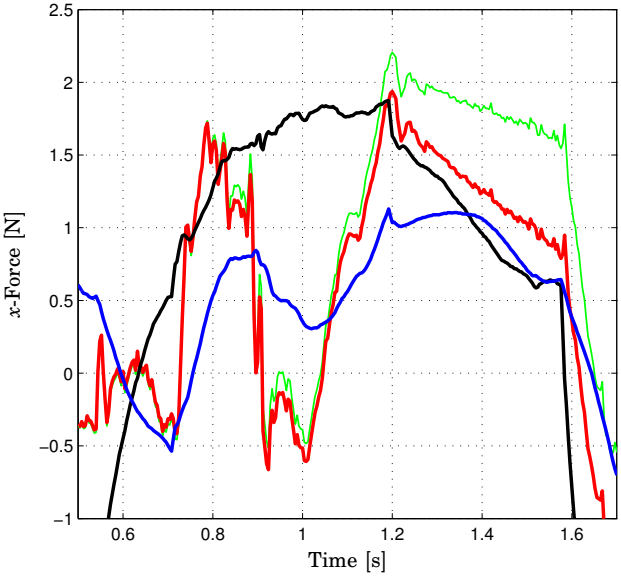

Figure 9.12 Selected part of the measured and estimated x-force around the transition from state 5 to 6.

controlled instead of the force. This changed strategy made the assembly less robust, but the effect was small concerning the uncertainties in this particular task. As the contact torque estimates were found out to be unreliable, the rotational search in state 7 was replaced with a position control 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 is a reasonable assumption to make, as the PCB is placed in a fixture. Gripping uncertainties, corresponding to small rotations around the $f2$ z-axis, were not expected to be a problem, as the gripper is 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. To be certain that the shield can was assembled correctly once the rotating motion was finished, the robot pressed the shield can with a large force towards the socket, kept

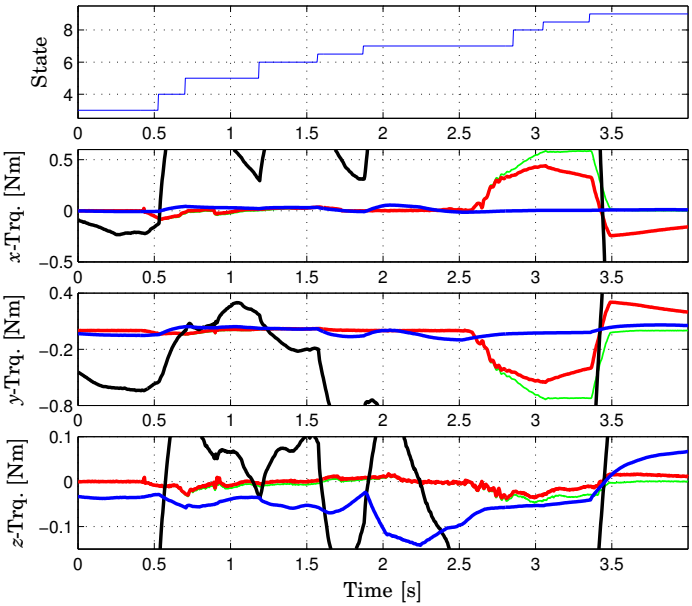

Figure 9.13 Measured and estimated torques from the the assembly sequence without force sensor together with the state sequence. Torques estimated with a priori information about the low torques are shown in blue, and torques estimated without a priori information in black. For reference, raw torque data from the force sensor is shown in green, and high-pass filtered torque data is shown in red. Some data has been cut away to show what happens in the measured torque. This shows that the black curve is a bad estimate.

the position of making contact for some time and then the assembly was assumed to be finished.

Force data from an experimental execution is given in Fig. 9.11. 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 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 is made with contact in both the z- and the y-direction, and this initially causes a friction peak in the x-force (at $t = 0.8$ [s]), the relevant part of the x-force is displayed in Fig. 9.12. The force then disappears and the contact is made at $t = 1.1$ [s]. The estimated force with a priori 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 is finally made at $t = 1.2$ [s]. The final search for the corner of the socket is 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 $t = 1.6$ [s] and the transition condition for the x-search at $t = 1.9$ [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 is rotated down onto the socket. The rotation is made around the origin of frame $f2$. Modeling errors in the position of this frame is the reason for the large z-forces around $t = 2.7$ [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 $t = 3.0$ [s]. Finally, the robot waited 0.3 seconds in state 8.5 and then moved away in state 9.

Measured and estimated torques from the experimental execution are given in Fig. 9.13. 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 estimated forces are reasonably correct when in contact, but the performance is worse when no contact is present, see, e.g, after $t = 3.4$ [s] in Fig. 9.11. The use of a priori information about the size of the external torques gives better force estimates, and they are in fact crucial for performing the assembly task considered, as one of the transition conditions only can be found using this estimate. The similarity between the high-pass filtered force data and the force estimate,

at least in the case when a priori information is used, verifies the high-pass character of the estimate. Most of the discrepancy between the measured and the estimated force is because of friction, as was previously described.

Screwing of nut

In this assembly scenario, torques were measured in frame $f2^a$. In this frame it is unlikely to get any significant torques around the x- and y-axes, but there will be a torque around the z-axis when the nut is tightened. This information was used for choosing R_F ($\bar{F}$ was chosen to be zero, as no bias force or torque were expected).

Experimental data from one execution is displayed in Fig. 9.14. No ground truth is available since the robot arms were not equipped with any force sensors. The first part was about putting the nut on the button. State 2 was a search in $f1^a$ z-direction, and the transition to the next state was a large z-force, which happened at $t = 1.6$ [s]. The distance between the arms was, however, not small enough, and therefore the robot moved back to a slightly modified start position and perfomed the z-search once more. This procedure was repeated two more times, and finally at $t = 10.2$ [s] the nut was finally put on the button.

The nut was then released and pushed down on the button, which can be seen as a z-force of 4 [N] at $t = 13.8$ [s]. Screwing was performed in state 6, and during this phase the side forces, x- and y-forces, were monitored. If their magnitude exceeded 3 [N] (illustrated with a dashed line), the robot would stop screwing. If it happened during the first two revolutions, it would release the nut, grip it again, and then continue screwing, but if it happened during the third or the fourth revolution, it would consider the nut to be tightened and the assembly finished. Such an event occurred during the fourth revolution of screwing, at $t = 25.9$ [s]. Also the estimated z-force was large, which might be an indication of a large torque. As it happened during the fourth revolution the assembly was finished. The entire sequence finishes with state 8 after approximately 26 [s] (18 [s] if the search for putting the nut on the button is excluded).

The screwing was made with a rotational velocity of 7 [rad/s] (400 [°/s]), and if the nut gets stuck on the thread or the robot gets a bad grip, large side forces appear quickly. The threshold therefore had to

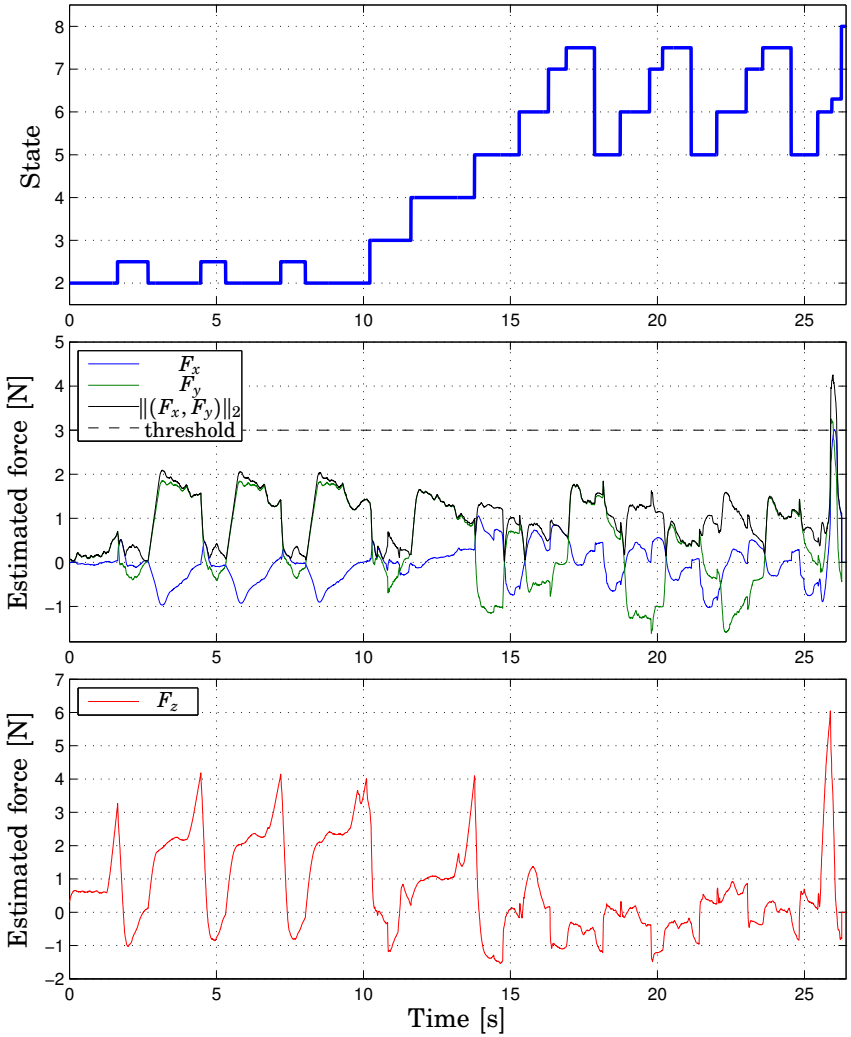

Figure 9.14 Experimental data from an execution of the nut screw scenario. The topmost diagram shows the state sequence (numbers according to Fig. 9.8). The middle diagram shows the estimated x- and y-force (in frame $f1^a$) together with their magnitude and the threshold used for detection of large side forces. The bottommost diagram shows the estimated z-force.

be quite low, to be able to react before the safety functionality of the robot reacted on too high torques in some of the joints.

The estimate of the z-torque is not shown, as this estimate was quite bad. The screwing was mainly made with the last joint of the robot arm holding the nut, and as the screwing was fast this generated a large control error for this joint, especially with detuned controllers. This gave a large bias in the estimation, and it was hard to see anything useful in the estimate.

9.6 Discussion

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

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 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, there may, however, 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 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 the nut screw assembly scenario both arms of FRIDA were used to perform a two-handed assembly, which meant that 14 joint errors was available to estimate the forces and torques, as compared to 7 joint errors in the single-armed case. This may have given better estimates, but it is hard to tell as no ground truth from a force sensor was available. 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.

9.7 Conclusions

A method for estimating the external forces acting on the end-effector of a robot has been described. It was based on the control errors for the low-level joint control loops. The method was experimentally verified in two small-part assembly tasks using a kinematically redundant robotic manipulator.

10

Conclusions

A framework for force controlled robotic assembly was presented. It was experimentally verified in different types of assembly tasks, with different types of robots. The framework can handle redundant manipulators, and assembly tasks in both single-arm and dual-arm configurations. The tasks are specified as constrained motions and they are modeled with finite state machines. The state transitions are based on measurements, either based on simple thresholds or more advanced classifiers based on the time evolution of the measurement signal, with the snap detection as one example.

It has been presented how uncertainties in assembly tasks can be managed. Geometric uncertainties were modeled by introducing uncertain transformations in the task description, and they were resolved by relating them to the measurements with a dynamic model. An example of uncertain gripping was resolved using a Kalman filter. Another type of uncertainty is the environmental properties. It was presented how the force controllers could be adaptively tuned based on identification of a contact model of the environment. 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 operators to specify and accomplish different types of assembly tasks.

A way to introduce a singularity-free orientation representation in the assembly framework was presented. By specifying the motions on the velocity level and keeping track of the actual orientation with a quaternion, it was possible to avoid the representation singularities

from the Euler angle representation. The method was experimentally verified in the execution of a peg-in-hole assembly task.

A method for estimating the force applied to the end-effector of the robot based on the joint control errors was presented. The method enables the use of force control in situations where an external force sensor is not present. The method was experimentally verified in two small-part assembly tasks, where the magnitude of the detected forces were of the same magnitude as the disturbances, mainly caused by friction in the joints.

A

Bibliography

ABB Robotics (2011): *Application manual - SoftMove*. Robot documentation M2004, rev E, RW5.14.

ABB Robotics (2012): *RAPID Reference Manual*.

ABB Robots (2012): `http://www.abb.com/product/us/9AAC100735.aspx`.

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. 7th IFAC Symp. Robot control (SYROCO)*, pp. 31–36. Wrocław, Poland.

Antonelli, G. and S. Chiaverini (1998): "Singularity-free regulation of underwater vehicle-manipulator systems." In *Proc. American Control Conf. (ACC)*, vol. 1, pp. 399–403. Philadelphia, USA.

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.

ATI (2012): `http://www.ati-ia.com/products/ft/sensors.aspx`.

Bishop, C. (2006): *Pattern recognition and machine learning*. Springer, New York.

Björkelund, A., L. Edström, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. Gestegård Robertz, D. Störkle, A. Blomdell, R. Johansson, M. Linderoth, A. Nilsson, A. Robertsson, A. Stolt, and H. Bruyninckx (2011): "On the integration of skilled robot motions for productivity in manufacturing." In *Proc. IEEE/ CIRP Int. Symp. Assembly and Manufacturing (ISAM)*, pp. 1–9. Tampere, Finland.

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

Blomdell, A., I. Dressler, K. Nilsson, and A. Robertsson (2010): "Flexible application development and high-performance motion control based on external sensing and reconfiguration of ABB industrial robot controllers." In *Proc. ICRA 2010 Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications*, pp. 62–66. Anchorage, Alaska, USA.

Bruyninckx, H. and J. De Schutter (1996): "Specification of force-controlled actions in the 'task frame formalism'-a synthesis." *Robotics and Automation, IEEE Transactions on*, **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.

Caccavale, F. and B. Siciliano (2001): "Quaternion-based kinematic control of redundant spacecraft/manipulator systems." In *Proc. Int. Conf. Robotics and Automation (ICRA)*, vol. 1, pp. 435–440. Seoul, Korea.

Cho, S., S. Asfour, A. Onar, and N. Kaundinya (2005): "Tool breakage detection using support vector machine learning in a milling process." *Int. J. of Machine Tools and Manufacture*, **45:3**, pp. 241–249.

De Schutter, J. (1988): "Improved force control laws for advanced tracking applications." In *Proc. Int. Conf. Robotics and Automation (ICRA)*, pp. 1497–1502. Philadelphia, USA.

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." *Int. J. Robotics Research*, **26:5**, pp. 433–455.

Di Lello, E., T. De Laet, and H. Buyninckx (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, Nevada, 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. Int. Conf. Robotics and Automation (ICRA)*, vol. 3, pp. 1588–1593. San Francisco, USA.

Eom, K., I. Suh, W. Chung, and S. Oh (1998): "Disturbance observer based force control of robot manipulator without force sensor." In *Proc. Int. Conf. Robotics and Automation (ICRA)*, pp. 3012–3017. Leuven, Belgium.

Erickson, D., M. Weber, and I. Sharf (2003): "Contact stiffness and damping estimation for robotic systems." *Int. J. Robotics Research*, **22:1**, pp. 41–57.

Freund, Y. and R. Schapire (1996): "Experiments with a new boosting algorithm." In *Proc. Thirteenth Int. Conf. Machine Learning*, pp. 148–156. Bari, Italy.

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.

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*, pp. 541–546. Chongqing, China.

Hacksel, P. and S. Salcudean (1994): "Estimation of environment forces and rigid-body velocities using observers." In *Proc. Int. Conf. Robotics and Automation (ICRA)*, pp. 931–936. San Diego, USA.

Haddadi, A. and K. Hashtrudi-Zaad (2008): "Online contact impedance identification for robotic systems." In *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 974–980. Nice, France.

Hamilton, W. (1840): "On a New Species of Imaginary Quantities, Connected with the Theory of Quaternions." In *Proc. Royal Irish Academy*, vol. 2, pp. 424–434.

Hogan, N. (1985): "Impedance control: An approach to manipulation: Parts i-iii." *ASME J. 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.

JGrafchart (2012): `http://www.control.lth.se/Research/tools/grafchart.html`.

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. Int. Conf. Robotics and Automation (ICRA)*, pp. 3687–3694. San Francisco, USA.

JR3 (2012): `http://www.jr3.com`.

Jung, J., J. Lee, and K. Huh (2006): "Robust contact force estimation for robot manipulators in three-dimensional space." *Proc., Part C: J. Mech. Eng. Science*, **220:9**, pp. 1317–1327.

Kalman, R. (1960): “A new approach to linear filtering and prediction problems.” *Trans. ASME–J. Basic Engineering*, **82:Series D**, pp. 35–45.

Khatib, O. (1987): “A unified approach for motion and force control of robot manipulators: The operational space formulation.” *Robotics and Automation, IEEE Journal on*, **3:1**, pp. 43–53.

Kock, S., T. Vittor, B. Matthias, H. Jerregård, 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)*, pp. 1–5. Tampere, Finland.

Kröger, T., B. Finkemeyer, M. Heuck, and F. Wahl (2004): “Adaptive implicit hybrid force/pose control of industrial manipulators: Compliant motion experiments.” In *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 816–821. Sendai, Japan.

Kyrki, V. (2008): “Quaternion representation for similarity transformations in visual SLAM.” In *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 2498–2503. Nice, France.

LabComm (2012): `http://wiki.cs.lth.se/moin/LabComm`.

Lefebvre, T., H. Bruyninckx, and J. De Schutter (2005): “Task planning with active sensing for autonomous compliant motion.” *Int. J. Robotics Research*, **24:1**, pp. 61–81.

Love, L. and W. Book (1995): “Environment estimation for enhanced impedance control.” In *Proc. Int. Conf. Robotics and Automation (ICRA)*, pp. 1854–1859. Nagoya, Japan.

Mallapragada, V., D. Erol, and N. Sarkar (2006): “A new method of force control for unknown environments.” In *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 4509–4514. Beijing, China.

Mason, M. T. (1981): “Compliance and force control for computer controlled manipulators.” *Systems, Man and Cybernetics, IEEE Transactions on*, **11:6**, pp. 418–432.

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)*, pp. 1–6. Tampere, Finland.

Natale, C., R. Koeppe, and G. Hirzinger (2000): "A systematic design procedure of force controllers for industrial robots." *Mechatronics, IEEE/ASME Transactions on*, **5:2**, pp. 122–131.

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*, pp. 670–675. San Diego, USA.

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*, pp. 1049–1054. Kobe, Japan.

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.

Raibert, M. and J. Craig (1981): "Hybrid position/force control of manipulators." *Journal of Dynamic Systems, Measurement, and Control*, **102:127**, pp. 126–133.

Real-Time Workshop (2012): `http://www.mathworks.se/products/simulink-coder/`.

Rethink Robotics (2012): `http://www.rethinkrobotics.com/index.php/products/baxter/`.

Rocco, P. and A. Zanchettin (2010): "General parameterization of holonomic kinematic inversion algorithms for redundant manipulators." In *Proc. Int. Conf. Robotics and Automation (ICRA)*, pp. 3721–3726. Anchorage, USA.

Rodriguez, A., D. Bourne, M. Mason, G. Rossano, and J. Wang (2010): "Failure detection in assembly: Force signature analysis." In *IEEE Conf. Automation Science and Engineering (CASE)*, pp. 210–215. Toronto, Canada.

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. Int. Conf. Intelligent Robots and Systems (IROS)*. Vilamoura, Portugal.

ROSETTA (2012): "Robot control for skilled execution of tasks in natural interaction with humans; based on autonomy, cumulative knowledge and learning." `http://www.fp7rosetta.org`.

Roy, J. and L. Whitcomb (2002): "Adaptive force control of position/velocity controlled robots: theory and experiment." *Robotics and Automation, IEEE Transactions on*, **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)*, pp. 376–381. Taipei, Taiwan.

Siciliano, B. and O. Khatib, Eds. (2008): *Springer handbook of robotics*. Springer.

Singla, P., D. Mortari, and J. L. Junkins (2005): "How to avoid singularity when using Euler angles?" *Advances In The Astronautical Sciences*, **119:II**, pp. 1409–1426.

Stateflow (2012): `http://www.mathworks.com/products/stateflow`.

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

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2012a): "Adaptation of force control parameters in robotic assembly." In *Proc. 10th IFAC Symp. Robot control (SYROCO)*. Dubrovnik, Croatia.

Stolt, A., M. Linderoth, A. Robertsson, and R. Johansson (2012b): "Robotic assembly using a singularity-free orientation representation based on quaternions." In *Proc. 10th IFAC Symp. Robot control (SYROCO)*. Dubrovnik, Croatia.

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 (2012): "Sensor-less compliance control for assembly robots." `http://www.toshiba-machine.co.jp/en/technology/tech_catalog/e3.html`.

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. Int. Conf. Robotics and Automation (ICRA)*, pp. 1108–1113. Shanghai, China.

Verscheure, D., I. Sharf, H. Bruyninckx, J. Swevers, and J. De Schutter (2010): "Identification of contact parameters from stiff multi-point contact robotic operations." *Int. J. Robotics Research*, **29:4**, pp. 367–385.

Weber, E., K. Patel, O. Ma, and I. Sharf (2006): "Identification of contact dynamics model parameters from constrained robotic operations." *ASME J. Dynamic Systems, Measurement, and Control*, **128**, pp. 307–318.

Wen, J. and K. Kreutz-Delgado (1991): "The attitude control problem." *Automatic Control, IEEE Transactions on*, **36:10**, pp. 1148–1162.

Willow Garage (2012): `http://www.willowgarage.com/pages/pr2/overview`.

Xenomai (2012): `http://www.xenomai.org`.

Xian, B., M. de Queiroz, D. Dawson, and I. Walker (2004): "Task-space tracking control of robot manipulators via quaternion feedback." *Robotics and Automation, IEEE Transactions on*, **20:1**, pp. 160–167.

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

Document name
LICENTIATE THESIS

Date of issue
December 2012

Document Number
ISRN LUTFD2/TFRT--3256--SE

Author(s)
Andreas Stolt

Supervisor
Rolf Johansson, Anders Robertsson

Sponsoring organisation
ROSETTA (EU FP7)

Title and subtitle
Robotic Assembly and Contact Force Control

Abstract

Modern industrial robots are traditionally programmed to follow desired trajectories, with the only feedback coming from the internal position/angle sensors in the joints. The robots are in general very accurate in tracking the desired motion, and they have become indispensable in many applications, such as spot welding and painting in the automotive industry. In more complex tasks, such as physical interaction with the environment, position control of the robot might be insufficient due to the fact that it is hard, or too costly, to achieve an environment that is structured enough. This is due to inherent uncertainties, such as part variations and inexact gripping.

One example of a challenging application is assembly, which is hard to accomplish using only position controlled robots. By adding a force sensor to the system, it gives the robot ability to correct for uncertainties by measuring contacts. This thesis presents a framework for force controlled robotic assembly. Assembly tasks are specified as sequences of constrained motions, where transitions are triggered by sensor events, coming either from thresholds or from more advanced classifiers. The framework is also able to explicitly deal with uncertainties, which can be estimated during execution to improve the performance. Further, a method for adaptation of force control parameters is presented, and how a singularity-free orientation representation can be used within the assembly framework. The case when no force sensor is available is also considered, and a method for estimating the external forces based on the joint control errors is presented. All methods presented are validated in experiments.

Key words
Robotics, force control, assembly, cooperating robots

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
122

Recipient's notes

Security classification