Constraint-based Task Specification and Estimation for Sensor-Based Robot Systems in the Presence of Geometric Uncertainty — Source link

Joris De Schutter, Tinne De Laet, J. Rutgeerts, Wilm Decré ...+4 more authors

Institutions: Katholieke Universiteit Leuven

Published on: 01 May 2007 - The International Journal of Robotics Research (SAGE Publications)

Topics: Robot kinematics, Mobile robot and Robot

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT WERKTUIGKUNDE
AFDELING PRODUKTIETECHNIEKEN, MACHINEBOUW
EN AUTOMATISERING
Celestijnenlaan 300B, B-3001 Heverlee (Leuven), Belgium

CONSTRAINT-BASED TASK SPECIFICATION AND ESTIMATION FOR SENSOR-BASED ROBOT TASKS IN THE PRESENCE OF GEOMETRIC UNCERTAINTY

Proefschrift voorgedragen tot het
bekomen van de graad van Doctor
in de Ingenieurswetenschappen

door

Johan RUTGEERTS

D/2007/7515/19
UDC 681.3*I29

31 mei 2007

D/2007/7515/19
ISBN 978-90-5682-785-4
UDC 681.3*I29

Voorwoord

“Zeg Johan, wanneer gaan die robots van jou nu eindelijk eens mijn afwas komen doen?” Het is onvoorstelbaar hoe vaak je dergelijke –weliswaar ludiek bedoelde– vragen te horen krijgt, als je doctoreert in de robotica. Blijkbaar roept de term ‘robot’ bij velen onmiddellijk connotaties op van intelligente, al dan niet mensachtige machines, die in staat zijn allerhande taken uit te voeren om het leven van de mens te vergemakkelijken. Deze visie wordt gesterkt door beelden van bijvoorbeeld de trompet spelende robots van Toyota, of de dansende robots van Sony[1]. Ik moet U echter teleurstellen: intelligente robots bestaan nog niet. Robottaken bestaan zo goed als altijd uit het letterlijk afspelen van voorgeprogrammeerde handelingen, binnen een exact gekende omgeving. De uitvoering van de taak ligt hierbij volledig vast, en kan niet aangepast worden aan eventuele –a priori ongekende– variaties in de omgeving van de robot. Met andere woorden: als zo’n dansende robot op een onvoorziene oneffenheid stapt, dan is de kans groot dat die robot omvalt, om dan al neerliggend verder te dansen omdat hij helemaal niet kan merken dat hij gevallen is. U begrijpt dus dat U voorlopig zelf nog zal moeten instaan voor Uw afwas.

Langs de andere kant vormt de idee van intelligente robotsystemen, die in staat zijn autonoom taken uit te voeren in hiertoe onaangepaste omgevingen, inderdaad de voornaamste drijfveer voor onderzoek in de sensorgebaseerde robotica. Bij sensorgebaseerde robotica worden robots uitgerust met sensoren, zoals een camera of een krachtsensor, om informatie in te winnen over hun omgeving. Door deze informatie op een correcte manier te interpreteren en de uitvoering van de taak aan te passen aan de actuele staat van de omgeving, kan een robot dan in principe autonoom zijn taak uitvoeren, ook al is de omgeving a priori niet gekend.

De huidige stand der techniek is nog ver verwijderd van echt intelligente robots, die taken kunnen uitvoeren in volledig ongestructureerde omgevingen, zoals een huiskamer. Voor taken in omgevingen met slechts een gedeeltelijke

[1]Online te vinden op Google of YouTube via de zoektermen ‘trumpet playing robot’ en ‘Sony dancing robot’ (25/04/2007).

geometrische onzekerheid, zoals bijvoorbeeld bij taken waarbij de objecten wel gekend zijn, maar hun posities en afmetingen slechts bij benadering, worden wel goede resultaten geboekt. In dit onderzoeksdomein is mijn doctoraat gesitueerd. Het beschrijft iTASC: een methodologie om dergelijke sensorgebaseerde taken te specificeren, met inbegrip van het schatten van ongekende geometrische parameters. Hoewel het –zoals zo vaak in onderzoek– slechts een kleine stap is naar meer intelligente robots, vormt de methodologie de basis voor de implementatie van verdere ondersteunende software, ter vervanging van COMRADE, de vorige-generatie specificatiesoftware van onze onderzoeksgroep.

Ik had het geluk te kunnen doctoreren in een hechte en gemotiveerde onderzoeksgroep, waarin de nadruk ligt op betrokkenheid en samenwerking tussen de collega's. Er zijn dan ook verschillende mensen die ik zou willen bedanken voor hun inbreng. Vooreerst gaat mijn oprechte dank natuurlijk uit naar Joris en Herman. Dankzij hen kreeg ik de mogelijkheid te doctoreren, en beiden zijn steeds bereid tijd vrij te maken voor hun doctorandi, om mee te discussiëren en te brainstormen. Joris en Herman hebben een fantastische onderzoeksgroep bijeengebracht: van de 'oude garde', Klaas, Tine, Peter en Walter, over 'mijn eigen generatie', Wim en Peter, tot de 'nieuwelingen', Tinne, Wilm, Ruben, Kasper en Diederik. Ik wil ieder van hen bedanken voor de vele discussies en de toffe sfeer binnen de groep. Ook Erwin en Brecht wil ik zeker bedanken voor de veelvuldige samenwerking. Verder zou ik graag Prof. Baeten en Prof. De Schreye willen bedanken voor hun begeleiding als assessor, en Prof. Duflou, Prof. Lefeber en Prof. Haegemans, om te zetelen in de jury voor mijn verdediging. Uiteindelijk wil ik graag mijn ouders bedanken voor de vele mogelijkheden en de brede opvoeding die zij mij gaven, mijn familie en vrienden voor alle leuke momenten samen, en natuurlijk ook Annelies, die mijn doctoraat van dichtbij meebeleefd heeft en mij steeds gesteund heeft tijdens mijn werk.

Johan Rutgeerts.
Leuven, 27 april 2007.

Abstract

Still today, robots in the industry are primarily used as positioning machines. A robot task in an industrial setting consist of replaying a fixed trajectory, and performing certain actions on specific positions along these trajectories. An example is robot spotwelding, in which a robot moves to a certain position to place a spotweld, then moves to another position to place the next spotweld, and so on. While executing such positioning tasks, the robot has no information about its environment. The robot executes its task blindly, and cannot adapt the task to variations in the position or the geometry of the objects with which it interacts. This implies that the environment of the robot must be fully adapted to the task. Currently, robot workcells are structured environments, specifically built for the robot to realize one single task.

The requirement to create a completely structured environment, adapted to each robot task, heavily restricts the number of applications in which robots can be used. Using robots for the automation of small product batches or batches with a high degree of product customization, is not economically viable. The relative cost of reprogramming the robot and adapting its environment to a new setting is too high, for production in small batches. Sensor-based robotics provides a possible answer to this problem. If a robot is equipped with extra sensors, such as a camera, force/torque sensor or laser distance sensor, the robot can measure its environment and adapt the task execution to the specific setting. During the last decades, sensor-based robotics has been an active research topic. The recently developed service robots such as automatic vacuum cleaners or robot lawn-mowers are the first commercial applications of this research. While nice applications of sensor-based robotics, these service robots have a very limited field of application. In research, more complex tasks have been realized, but very few applications have been adopted in industry. The main reason for this is the complexity of the sensor measurement processing, and the lack of a true task specification formalism for complex sensor-based tasks.

This thesis presents *iTASC (instantaneous Task Specification using Constraints)*, a systematic and generic constraint-based methodology to specify sensor-based robot tasks, including support for the estimation of geometric parameters. By estimating these geometric parameters, the robot task can be adapted to variations in the geometry or in the pose of the objects in the robot's environment. The methodology introduces the concepts of *objects* and *features* of those objects, relevant to the robot task. The objects and features are represented by frames, and their motion modeled in terms of a set of coordinates, called *feature twist coordinates*. A task is then specified by defining constraints on the feature twist coordinates. Instantaneous optimization techniques are used to solve these constraints for the motion of the robot. To estimate geometric parameters, extra uncertainty coordinates are introduced in a similar way as the feature twist coordinates. In every timestep, the pose of the object and feature frames must be known. This thesis presents a systematic way to calculate these poses, based on the information in the pose closure equations. Finally, multiple example tasks are presented. They illustrate the practical application of the specification methodology.

Beknopte Samenvatting

De huidige-generatie industriële robots worden voornamelijk gebruikt als positioneermachines, en realiseren een taak door het afspelen van een vooraf vastgelegd traject, waarbij op bepaalde plaatsen langs dat traject een actie wordt uitgevoerd. Een voorbeeld van zo'n robottaak is het gerobotiseerd puntlassen. In deze taak beweegt de robot naar een bepaalde positie en plaatst daar een puntlas, daarna beweegt de robot naar een nieuwe positie voor de volgende puntlas, en zo verder. Bij het uitvoeren van een dergelijke taak beschikt de robot niet over informatie over zijn omgeving. De robot voert zijn taak blindelings uit en kan dus geen rekening houden met variaties in de pose of de geometrie van de objecten waarmee hij interageert. Dit impliceert dat de omgeving van de robot volledig aangepast moet zijn aan de robottaak. Een robotcel is daarom een gestructureerde omgeving, volledig toegespitst op het realiseren van één bepaalde taak.

De vereiste om een volledig gestructureerde robotomgeving te creëren beperkt de inzetbaarheid van robots sterk. Het gebruik van robots om kleine productseries of productseries die regelmatig veranderen te automatiseren, is economisch niet haalbaar. De kost om de robot telkens te herprogrammeren en zijn omgeving aan te passen aan de nieuwe taak is relatief gezien te hoog, indien geproduceerd wordt in kleine aantallen. Sensorgebaseerde robotica biedt een mogelijke oplossing voor dit probleem. Door een robot uit te rusten met sensoren zoals een camera, een krachtsensor of een laser-afstandssensor kan de robot zijn omgeving waarnemen en de uitvoering van de taak aanpassen aan de specifieke toestand van de omgeving. Reeds vele jaren vormt de sensorgebaseerde robotica een belangrijk onderzoeksonderwerp. De recent ontwikkelde servicerobots voor taken zoals het stofzuigen van een huiskamer of het maaien van een grasveld, zijn de eerste commercieel beschikbare toepassingen van dit onderzoek. Het zijn mooie voorbeelden van sensorgebaseerde robotica, maar telkens toegespitst op –en beperkt tot– een zeer specifieke applicatie. Hoewel complexere robottaken reeds gerealiseerd zijn in onderzoeksomgevingen, is er slechts een heel geringe doorstroom naar echte industriële toepassing. De hoofdreden voor deze geringe doorstroom zijn de complexiteit van de interpretatie van de sensormetingen en het ontbreken van een ondersteuning voor de taakspecificatie van complexe sensorgebaseerde taken.

Dit proefschrift bouwt voort op de jarenlange ervaring in sensorgebaseerde robotica aan onze onderzoeksgroep en presenteert *iTASC (instantaneous Task Specification using Constraints)*, een systematische methodologie voor de ogenblikkelijke specificatie van sensorgebaseerde robottaken met inbegrip van geometrische schattingsproblemen. Door geometrische parameters te identificeren kan een robottaak aangepast worden aan variaties in de geometrie of pose van objecten in de omgeving van de robot. De methodologie maakt een onderscheid tussen *objecten* en *kenmerken* van die objecten, die relevant zijn voor de uit te voeren taak. De objecten en kenmerken worden voorgesteld door assenstelsels, en hun bewegingen worden gemodelleerd als een stel van coördinaten, die *kenmerk-twistcoördinaten* worden genoemd. Een taak wordt uiteindelijk gespecificeerd door beperkingen te definiëren op de kenmerk-twistcoördinaten. Ogenblikkelijke optimalisatietechnieken worden aangewend om de robotbeweging te bepalen die voldoet aan de beperkingen. Om geometrische parameters te schatten worden extra onzekerheidscoördinaten ingevoerd, op een gelijkaardige manier als de kenmerk-twistcoördinaten. In elke tijdstap moeten de poses van de object- en kenmerkassenstelsels gekend zijn. Dit proefschrift stelt een systematische manier voor om deze poses te berekenen, gebaseerd op de informatie in de pose kringloopvergelijkingen. Verder worden verschillende voorbeeldtaken uitgewerkt, die het praktische gebruik van de methodologie illustreren.

Symbols, definitions and abbreviations

General abbreviations

1D, 3D, 6D	: 1-, 3-, or 6-dimensional
DOF	: degree of freedom
ACM	: Autonomous Compliant Motion
OROCOS	: Open RObot COntrol Software
iTASC	: instantaneous Task Specification using Constraints
DES	: Discrete Event System
TFF	: Task Frame Formalism
KF	: Kalman Filter
EKF	: Extended Kalman Filter
IEKF	: Iterated Extended Kalman Filter
NMSKF	: Non-Minimal State Kalman Filter

General symbols and definitions

a	: scalar (unbold lower case)
$\boldsymbol{a}$	: vector (bold lower case)
$\boldsymbol{A}$	: matrix (bold upper case)
$\dot{a}$	: time derivative of a
$\|\|a\|\|$	: Euclidean norm of a
$\|\|a\|\|_{\boldsymbol{W}}$	: weighted norm with weighting matrix $\boldsymbol{W}$
$\boldsymbol{A}^{\dagger}$	: Moore Penrose pseudo-inverse
$\boldsymbol{A}^{\#}$	: weighted pseudo-inverse
$[\boldsymbol{p}\times]$	: matrix expressing the cross product with $\boldsymbol{p}$
t	: time
$\boldsymbol{z}$	: sensor measurement
$\boldsymbol{K}_{st}$	: stiffness matrix
$\boldsymbol{J_C}$	: camera Jacobian

$\widetilde{x}$	: a prediction for x
$\widehat{x}$	: an estimate for x

Rigid body kinematics

$\boldsymbol{d}_a^b$	: finite displacement of object b with respect to a
$\boldsymbol{T}_a^b$	: 4×4 homogeneous transformation matrix of frame b with respect to a
$\boldsymbol{R}_a^b$	: 3×3 rotation matrix of frame b with respect to a
${}_c\boldsymbol{p}^{a,b}$	: position vector from a to b, expressed in frame c
x, y, z	: translational coordinates
ϕ, θ, ψ	: rotational coordinates
$\mathbf{t}$	: twist
${}^c_d\mathbf{t}_a^b$	: twist of b with respect to a, with reference point c and reference frame d
${}_f\boldsymbol{v}$	: translational velocity vector, expressed in frame f
${}_f\boldsymbol{\omega}$	: rotational velocity vector, expressed in frame f
${}^f_g\boldsymbol{P}$	: screw projection matrix from f to g
$\boldsymbol{M}_b^a$	: reference point transformation matrix from a to b
$\boldsymbol{E}$	: 3×3 matrix transforming the time rates of an angular representation into rotational velocities $\boldsymbol{\omega}$
$\boldsymbol{\mathcal{E}}$	: 6×6 matrix transforming a twist into the time rate of a finite displacement

iTASC

$\boldsymbol{q_R}$	: robot joint positions
$\dot{\boldsymbol{q}}_{\boldsymbol{R}}$	: robot joint velocities
$\boldsymbol{J_R}$	: robot jacobian
$\boldsymbol{e}(\boldsymbol{q_R}, t)$	: a Task Function
$o1$	: frame attached to object 1
$o2$	: frame attached to object 2
$f1$	: frame attached to feature 1
$f2$	: frame attached to feature 2
$\boldsymbol{J_F^a}$	: the feature Jacobian for feature a
$\boldsymbol{\tau}^a$	: the feature twist coordinates for feature a
$\boldsymbol{J_{FI}}$	: the feature Jacobian for submotion I
$\boldsymbol{J_{FII}}$	: the feature Jacobian for submotion II
$\boldsymbol{J_{FIII}}$	: the feature Jacobian for submotion III
$\boldsymbol{\tau}_I$	: the feature twist coordinates for submotion I
$\boldsymbol{\tau}_{II}$	: the feature twist coordinates for submotion II
$\boldsymbol{\tau}_{III}$	: the feature twist coordinates for submotion III

$\boldsymbol{C_R}$	: linear coefficient matrix for constraints on $\boldsymbol{q_R}$
$\boldsymbol{C_F}$	: linear coefficient matrix for constraints on $\boldsymbol{\tau}$
$\boldsymbol{J}_s$	: sensor Jacobian
$\boldsymbol{\chi_U}$	: uncontrolled or unknown degrees of freedom
$\boldsymbol{\chi}$	: feature pose coordinates
$\boldsymbol{\chi}_I$	: feature pose coordinates for submotion I
$\boldsymbol{\chi}_{II}$	: feature pose coordinates for submotion II
$\boldsymbol{\chi}_{III}$	: feature pose coordinates for submotion III
$\boldsymbol{H_R}, \boldsymbol{H_F}, \boldsymbol{H_U}$	: coefficient matrices of a linearized measurement equation
$\boldsymbol{J_U}$	: Jacobian for the unknown degrees of freedom

Table of contents

List of Figures

Chapter 1

Introduction

1.1 Background

Since their introduction in the 1960's, robots have become widely accepted as an indispensable means to increase productivity and automation in manufacturing environments. While industrial robots are still most commonly used in the automotive industry (with in Japan, Italy and Germany more than 1 robot per 10 production workers[1]), new robot technologies are emerging in other sectors such as the aerospace, chemical and medical industry. During the last decade, robot manufacturers have seen a worldwide increase in robot sales, with average annual growth rates of about 7%. The rapid decline of robot prices, combined with the increase of both their quality and the labor costs, are the main reasons for this steady growth.

Robots are very flexible machines, and seem ideal to automate parts of a production process. They can indeed be used for numerous tasks, such as pick-and-place tasks, spot and arc welding, or spray painting. An example of such a task is given in Figure 1.1. This figure shows a fully automated robot station performing adhesive and welding operations to connect the side frame of a car with its body shell. Robots are ideal for such tasks because they are fast, accurate, and reliable. Furthermore, robots work around the clock and withstand harsh or hazardous environments. On the other hand, robots also have limitations. Robots are capable of a lot of tasks, but they do their work blindly, without intelligence or knowledge about their environment. Robots usually have no other sensors than their internal encoders. Because of this, robots only know how they are positioned, but not what happens in their environment. For each robot application, specific peripheral equipment is needed to structure the environment. Only when all workpieces

[1]Source: United Nations Economic Commission for Europe. http://www.unece.org/

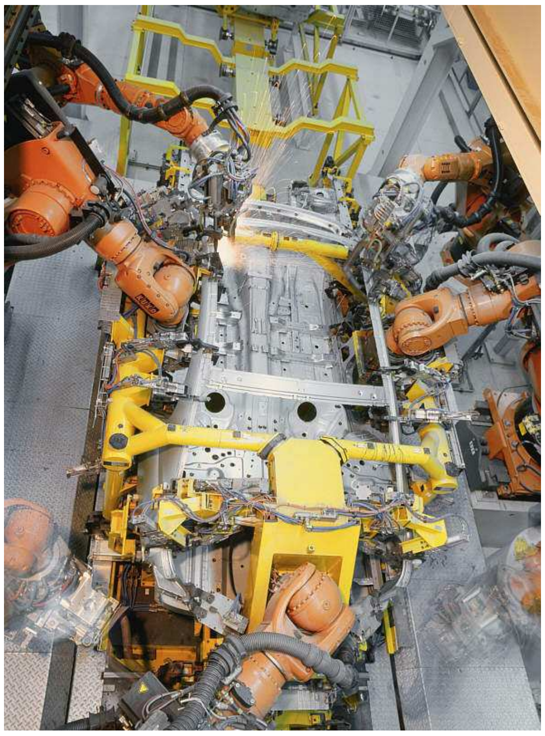


Figure 1.1: A fully automated robot station doing adhesive and welding operations to connect the side frame of a car with its body shell. Besides the robots, the peripheral equipment can be seen, which holds the car panels together while they are being welded or glued.

are at an exactly known position, a robot can realize its task by replaying a set of actions on preconfigured positions or along preconfigured trajectories. The need to change the robot environment according to each new task for the robot, makes the implementation of a robotic application a difficult challenge. Furthermore, the cost of the peripheral equipment, as well as the cost of programming a robot, make up for a substantial part of the complete cost of a robot cell. As each specific application requires custom peripheral equipment and a custom control program, a robot is not cost effective if frequent changes in the production process are needed. This prevents a lot of potential robot automation projects from being realized, as these projects are not economically feasible in companies which produce small and customized batches.

A possible solution to the high cost of a structured robot environment is to equip a robot with more sensors than just its internal encoders, such as a force/torque senor, a camera or a laser distance sensor. These extra sensors provide information about the robot environment and allow the identification of the objects in the environment and their positions. Using the sensor measurements, the robot can perform the task without knowing its environment in advance. This opens up possibilities to use robots in less structured settings, and lowers the prerequisites of the peripheral equipment. However, while research has been actively focusing on sensor-based robotics for the last decades, the ultimate goal of intelligent and autonomous robots still remains an open topic, and few applications of sensor-based robotics have been put to practice in true industrial settings.

1.2 Research at our group

Previous work at our research group has always had a strong focus on sensor based robotics. Our research aims towards more flexible and intelligent robots, capable of all sorts of tasks in unstructured or dynamic environments. Historically, research at our group primarily focused on force controlled *autonomous compliant motion* (ACM) and *task specification* for autonomous compliant motion tasks. More recently, the focus shifted to task specification for *general sensor-based tasks*, and *estimation of geometric parameters* during task execution.

Autonomous compliant motion

Force controlled compliant motion tasks (De Schutter and Van Brussel 1988a) are tasks in which a robot makes contact with its environment to realize a task. Examples of such compliant motion tasks are assembly, manipulation, deburring and milling. *Autonomous* compliant motion tasks are compliant motion tasks in which the robot has a certain degree of autonomy, in that it can still perform its task when there is geometrical uncertainty in the robot environment. For instance, when deburring castings, not all castings have exactly the same size, they are not necessarily always placed at the same position with respect to the robot, and the burrs can be at different places and have different shapes. To cope with this uncertainty, the robot is equipped with a force/torque sensor. This sensor measures the contact forces and torques between the robot (or the robot tool) and the object it makes contact with. The task execution is continuously adapted according to these measurements, to adequately perform the task even when the modeled position or geometry of the object is not the same as the real position or geometry.

Research in ACM at our research group culminated with the development of COMRADE (Van de Poel et al. 1993), a specification software package to

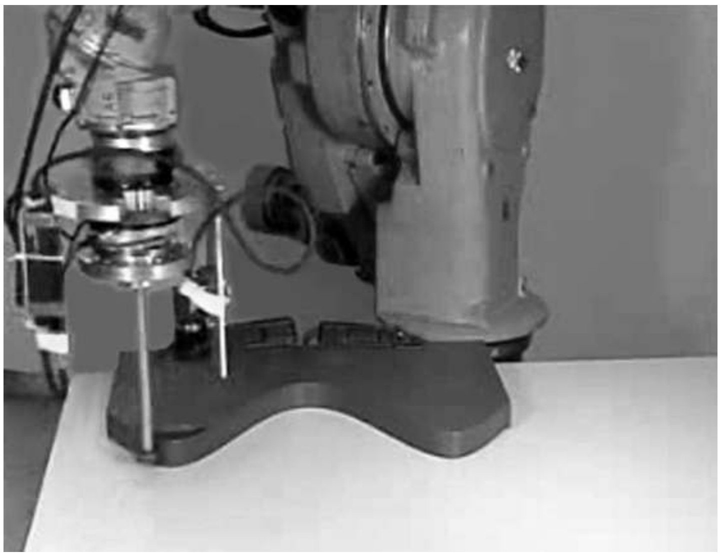


Figure 1.2: A robot performing a contour following experiment, in which both force measurements are used in a feedback loop, as well as camera images to generate a feedforward signal.

specify ACM tasks. Using COMRADE, it is possible to specify ACM tasks in a script form, by specifying target contact forces or velocities for the robot. An example of a robot task specified using COMRADE is shown in Figure 1.2. The task shown in this figure is a contour following task, in which both force measurements are used in a feedback loop, as well as camera images to generate a feedforward signal (Baeten and De Schutter 2003).

Estimation of geometric parameters

Another important research topic at our group is the estimation of geometric parameters, such as the position or the geometry of objects in the robot environment. Instead of using sensor measurements directly in the control loop, Bayesian techniques are used in an intermediate interpretation step, for the purpose of model building or model identification. The most commonly used techniques are the Kalman filter variants (Kalman 1960), such as the extended Kalman filter (Tanizaki 1996) or the non-minimal state Kalman filter which was developed in our group (Lefebvre, Bruyninckx, and De Schutter 2005b). More recently, also particle filters (or sequential Monte Carlo methods) were used (Doucet, Gordon, and Krishnamurthy 2001), as these are more appropriate for highly non-linear estimation problems. Moreover, particle filters can handle estimation problems with hybrid states, that is, states which consist of both continuous and discrete variables. The drawback of particle filters is their computational complexity. Particle filters generally have much higher computational demands than Kalman filters, which makes them less appropriate for high-dimensional estimation problems.

OROCOS: Open Robot Control Software

In late 2000, the Orocos project (www.orocos.org) was initiated at our research group. Orocos stands for Open RObot Control Software, and is a modular and architecture independent software framework for real-time robot and machine control. Orocos consists of four sub-projects:

- The **RealTime Toolkit** (RTT) is a library for realtime control services. It is a framework providing the building blocks to create robot or machine control programs.
- The **Kinematics and Dynamics Library** (KDL) provides modeling support for robot (or general machine) kinematics and dynamics.
- The **Bayesian Filtering Library** (BFL) is a framework to implement Bayesian estimators. The library includes support for different Kalman filter variants, particle filters and Kalman smoothers, but is easily extensible to include other Bayesian methods.
- The **Orocos Components Library** (OCL) contains components which are built using the Realtime Toolkit, and which can be used as (a part) of a robot or machine control program.

All the software of the Orocos project is available as open source software, released under the LGPL license.

1.3 Motivation and Contributions

Widespread application of sensor-based robotics is only possible if sensor-based tasks can easily be specified. To support the task specification of those tasks, several specification methodologies were developed in robotics research, such as the Task Frame Formalism (De Schutter and Leysen 1987), or a number of methodologies based on the definition of instantaneous twist and wrench spaces (Aghili 2005; Liu and Li 2002).

While these approaches are very useful for certain kinds of tasks, they have a number of common disadvantages. Only simple geometric models are used to support the task specification, such as a single task frame or a single twist and wrench space. This means that these methodologies are inadequate to specify tasks which comprise control constraints from different sources, such as different sensors yielding partial information to realize the task. Indeed, apart from few exceptions (Baeten and De Schutter 2003; Kröger, Kubus, and Wahl 2006), the approaches focus on tasks in which a single sensor is used. Also for instance tasks involving partial specifications for different parts of a robot, cannot be handled. Such tasks require a true constraint-based approach, in which different motion or sensor constraints can be specified independently. Furthermore, current specification methodologies provide little to no support for the estimation of geometric parameters.

This thesis presents iTASC (instantaneous Task Specification based on Constraints): a methodology to specify sensor-based robot tasks. iTASC contributes to the state of the art on the following aspects:

- **its constraint-based nature:** iTASC is a true constraint-based methodolo-gy. As a result, iTASC is suitable to specify tasks which comprise control constraints from different sources. In iTASC, tasks are specified according to the *Task Function Approach* (Samson, Le Borgne, and Espiau 1991). To support the expression of the Task Function, the concepts of *objects* and *features* are introduced, and constraints defined on the relative motion of these objects and features. The methodology allows tasks to be fully, under or overconstrained.

- **its generic nature:** iTASC is a generic approach, in that it does not focus on one particular kind of task, sensor or robot system. iTASC can be used for general, velocity resolved robot systems consisting of rigid links and joints. Also, all sensors yielding geometric information can be used (for instance, cameras or distance sensors, but also dynamical sensors such as a force sensor, if the task is executed in a quasistatic way).

- **the model update procedure:** All task specification methodologies use a geometric model of the task, which must be kept up to date while executing the task. While other task specification approaches require the programmer to implement ad-hoc procedures to this end, iTASC provides a generic update procedure to automatically track the poses of the object and feature frames of the model.

- **the integration of estimation:** Where other methodologies rely on the intrinsic robustness of sensor-based control to deal with geometric uncertainty, iTASC provides the means to explicitly model and estimate uncertain parameters.

To illustrate the effectiveness of the approach, the iTASC methodology is applied to the following example applications:

- A minimally invasive surgery task is used as an example to introduce the concepts of iTASC (Section 4.2, page 38). Though it is a pure positioning task, in which no extra sensors are used, it is still a nice example of a task which is not easily specified using a single task frame, as there are two sources of constraints. Section 6.2 (page 81) shows how this application can easily be extended to a different setting, in which the laparascopic tool used in the task has two extra degrees of freedom.
- The forming task, discussed in Section 6.3 (page 85), is an example of a fully constrained task involving a force controlled contact between curved surfaces.
- The inspection task in Section 6.4 (page 89) has similar constraints as the minimally invasive surgery task, but is underconstrained and requires state transitions (that is, transitions between different control constraints).
- The incompatible seam following task (multi-point contact) is a typical example of a task which cannot easily be specified according to methodologies such as the Task Frame Formalism. Section 6.4 (page 89) shows that it is straightforward to specify this task according to iTASC.
- The laser engraving task in Section 6.6 (page 100) is an example of an underconstrained task, involving two different sources of constraints, and estimation of geometric parameters.
- The final example application deals with human-robot shared control (Section 6.7, page 108). This task uses multiple sensors to allow a human operator and a robot to work together to realize a task. The choice of the weights for the different control constraints defines the dynamic behavior of the task.

1.4 Outline of this thesis

This thesis is organized as follows:

- Chapter 2 gives an overview of historic and recent research progress in robotics, with emphasis on task specification for ‘sensorless’ robot tasks, this is, without additional sensors, as well as ‘sensor-based’ tasks, this is, with additional sensors, such as a force/torque sensor or a camera.
- Chapter 3 introduces some mathematical concepts and notations, which are assumed known in the further chapters of this thesis.

- Chapter 4 presents the core of iTASC: a systematic procedure to model and specify sensor-based tasks. Inspired by (Ambler and Popplestone 1975), objects and features are introduced. The objects and features are represented by frames, and their motion expressed in terms of a set of coordinates, called feature twist coordinates. A task is then specified by defining constraints on the feature twist coordinates.

- In every timestep, the pose of all object and feature frames must be known. While the programmer can implement an ad-hoc procedure to calculate these frame poses, generic support is desirable as this lessens the work involved when specifying a task, and opens up possibilities to develop task specification support software which automates the task specification process as much as possible. The model update procedure of iTASC is presented in Chapter 5.

- Whereas most previous work relies on the intrinsic robustness of sensor based control to deal with geometric uncertainty, the approach presented in this thesis explicitly models geometric uncertainty. To this end an additional set of uncertainty coordinates is introduced. A generic scheme is proposed to estimate these coordinates and to take them into account in the control loop, also in Chapter 5.

- Simulation and experimental results of several example applications are presented in Chapter 6.

- Finally, Chapter 7 gives an overview of the main contributions and conclusions of this thesis, an evaluation of the limitations of the formalism and gives suggestions for future research topics.

Chapter 2

Literature Survey

2.1 Introduction

This chapter gives an overview of historic and recent research progress in robotics, with emphasis on task specification for 'sensorless' robot tasks, this is, without additional sensors, as well as 'sensor-based' tasks, this is, with additional sensors, such as a force/torque sensor or a camera.

First, Section 2.2 gives a conceptual overview of the components which constitute a robot control program. Section 2.3 then discusses task specification support, this is, different approaches which aim to aid the programmer in specifying a robot task, by providing a functional implementation of (some of) the components identified in Section 2.2. Using such task specification software, specifying a robot task is reduced to filling in task-specific parameters of the available components. Section 2.4 discusses further relevant research effort, which however does not focus on the whole of task specification but on the estimation component individually.

2.2 Robot control programs

In se, a robot is nothing more than a few lumps of iron, interconnected by gears and actuators, and to some extent equipped with sensors. Only with the correct robot control program, sending the appropriate control signals to the actuators, a robot becomes a useful system capable of realizing tasks. So, for each robot task, a control program is needed. During the last decades, robotics research has seen an evolution in the complexity of these control programs. The first robots were programmed manually, and on the robot controller itself. These controllers provided only simple programming primitives, such as joint space and Cartesian movements, and hence only allowed for simple tasks.

Recent robotics applications are implemented on a lot more complex systems: more complex *types of robotic systems* are used, such as humanoid, parallel or cooperating robots, *sensing and estimation* is used to adapt the task execution online to variations of the robot and its environment, and robots are used in *different environments*, for which they were not necessarily designed for, such as a household environment. Hence, the robot control programs have become a lot more extensive and complex.

Figure 2.1 shows a conceptual scheme of the components needed to realize a robot task. It is an extended version of the traditional three-layer architecture for intelligent control (Saridis 1979). While only complex robot tasks use all these components, the general structure is valid for all robot tasks. In the case of simple tasks, several of the components can be empty.

Central in the scheme is the actual control program, which takes operator commands and sensor data as an input and generates actuator setpoints as its output. Each robot task consists of a number of states. While these might be as simple as *ready to start*, *running* and *finished*, for tasks of little complexity, more elaborate tasks have a more complex state diagram. Hence, in every control program a **discrete event system** (**DES**) or finite state machine, reflecting this state diagram, is more or less prominently present. It reacts to operator commands (such as *start* or *stop*) and to events from the robot system or from the running control components (such as an emergency stop event). Based on these commands and events, the DES deliberates possible state transitions and selects an active state. Furthermore, according to this active state the DES selects a set of control components which realize the desired behavior of the robot during that state.

The control components are the functional blocks, which process the sensor data from the robot system, and generate the actuator signals. There are active and passive control components. The active components, denoted by a rectangular box in the scheme, have a certain behavior, this is, they generate data such as setpoints or control outputs. The passive components, denoted by a rounded box in the scheme, are data containers holding the data from the active components, as well as possible other, contextual information, as for instance the kinematic and dynamic models of the task.

Central is the **model**, which is used by the active components: the **path generator**, the **estimator** and the **controller**. The model comprises all information needed for the calculations of the active components, such as kinematic and dynamic models, or sensor models yielding measurement equations. In other words, the model provides the necessary *context* for the active components.

The content of the model is both used as well as updated by the active components. This is most clear for the estimation component, which has this as its sole purpose: it uses model data such as a measurement equation to

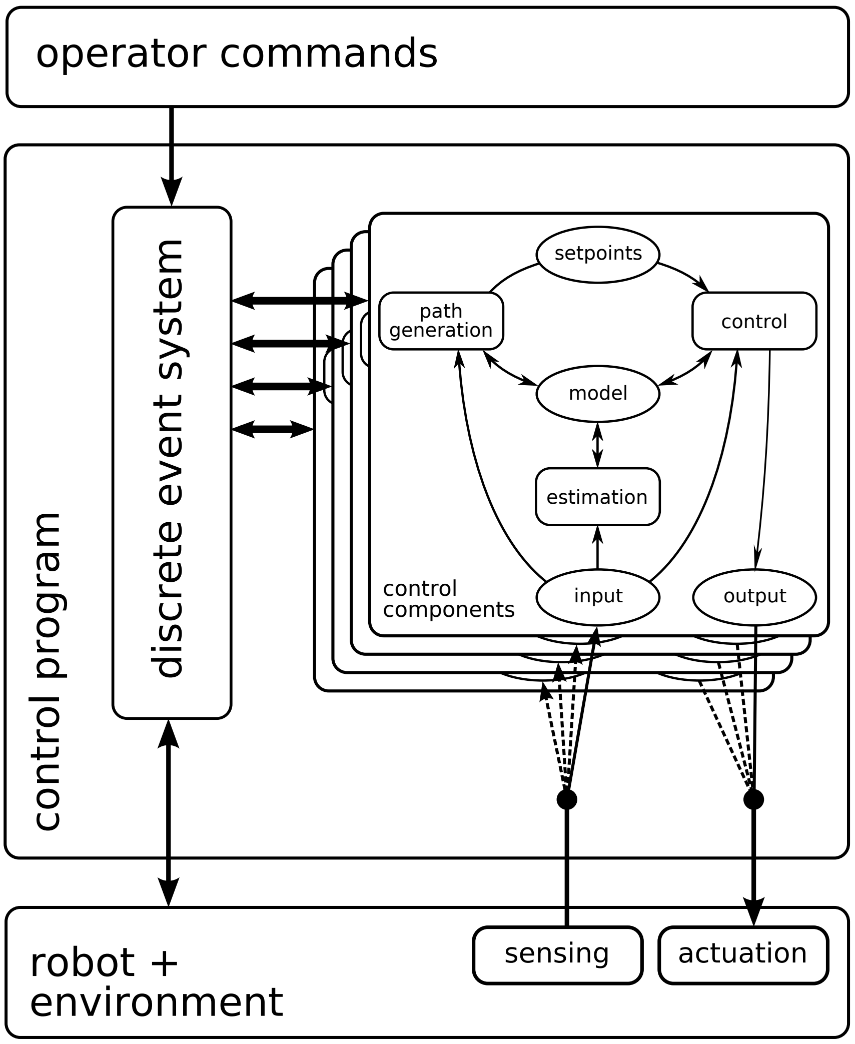


FIGURE 2.1: The components constituting a robot application. Central is the control program. It contains a discrete event system, which deliberates transitions between different states of the task, based on operator commands, events from the robot system and events from the running control components. According to the state of the task, the discrete event system selects a specific set of control components which realize the desired behavior of the robot during that state. Active components, this is, which generate data, are denoted by a rectangular box. Passive components, this is, data containers holding data from active components as well as possibly other information, are denoted by a circular box.

estimate model parameters from its **inputs**, the sensor data coming from the robot system. Hence, it uses, as well as updates the model. To a lesser extent this is also valid for the other active components, the path generator and the controller. The path generator generates the **setpoints** for the controller, and the controller uses these setpoints to calculate actuator **outputs**. Although both the path generator and the controller use model data for this purpose, they possibly also update the model. For instance, for a specific application, the controller could update the model with information about the tracking error, which is then used by the estimator to calculate a better estimate of certain geometric parameters of the model. This reciprocal data flow is indicated by the double arrows in the scheme.

A specific robot task is realized by the set of all these components, this is, the DES and, for each state of the application, a different path generator, estimator and controller, as well as their own input, output, setpoint and model data containers.

2.3 Task specification

To specify a robot task, all components of the scheme of Figure 2.1 should be filled in by the programmer. While the scheme is conceptually very simple, implementing a complete robot control program, including all the components, from scratch is a very elaborate task as each of the components can have extensive inner operations. Hence, several approaches have been presented to aid the programmer in specifying a task. Each of these approaches provides ready-to-use implementations for several of the components, hereby relieving the programmer to some extent by reducing the programming effort to filling in the remaining components, or even to filling in only a number of parameters of otherwise fully functional components.

This section gives an overview of such task specification approaches. A distinction is made between *sensorless* tasks, for which the robot is used as is, without extra sensors, and *sensor-based* tasks, in which extra sensors are used such as a force sensor or camera. This division is artificial; for instance, task level programming (discussed in Section 2.3.1) actually focuses on all tasks in general, without making the distinction between sensor-based and sensorless. Nevertheless, at least two reasons justify this choice. Firstly, most robots in industry are still used for purely geometric tasks without extra sensors, in contrast to research, which extensively has focused on sensor-based robotics during the last decades. Secondly, general support for task specification is a lot less common for sensor-based tasks, due to the additional complexity compared to sensorless tasks.

2.3.1 Sensorless tasks

Initially, robot controllers were self-contained systems without external computers or other support hardware, and task specification was hence limited to the methods these controllers provided. Specifying a task consisted of the definition of viapoints for the robot motion, the type of motion between those viapoints (for instance, joint space or Cartesian motion), and possibly an action at each viapoint (such as opening or closing a gripper). The viapoints were specified offline, as a list in a control program, or taught online by moving the robot to the desired positions. Three ways to do this teaching can be distinguished:

- *walk-through teaching:* In walk-through teaching, a teach pendant is used to move the robot into the viapoint positions (Marcelo, Lin, and Lim 1999).
- *lead-through teaching:* In lead-through teaching, viapoints are specified by physically interacting with the robot to move it into the desired positions (Todd 1986). For non-backdrivable robots, this requires active sensing and control of the robot to follow the operator's input.
- *teleoperation:* In teleoperation, a master-slave setup is used to lead the robot along the desired path. The robot (the slave) is controlled to follow the input of a joystick or haptic device (the master).

Further programming primitives provided by these controllers are simple geometrical primitives such as lines and arcs, similar to what is present in the controllers of computer numerically controlled (CNC) machines, such as milling machines or lathes.

The control components of Figure 2.1 are easily recognizable for these kind of applications: controllers, typically PID, are provided for Cartesian and joint space motion, the model is given by the kinematics of the robot (or, for joint space tasks, no model is needed), and path generators are present to interpolate between the viapoints in joint or Cartesian space, and to generate the setpoints for motion along the provided geometric primitives. The only inputs to the control program are the joint positions, and, as most industrial robot controllers have joint velocity controllers in hardware, the outputs are typically given by voltage setpoints for these controllers, proportional to the desired joint velocities. There is no estimator, and the DES only contains the logic to switch generators and controllers according to the desired motion, and to deal with very few states, such as *initialized*, *running* or *in error*.

While the set of programming primitives provided by these robot controllers is not very extended, it proves adequate for a large number of industrial tasks, such as positioning tasks, pick-and-place tasks or trajectory following, in well structured environments. Still today, a large number of robots in use are being programmed that way. On the other hand, especially in research, the desire is to use robots for more complex tasks and in less structured environments, such as domestic and service environments. These kinds of tasks are impossible to program using the programming primitives of current robot controllers.

Complex Task Kinematics

Tasks involving complex geometries, such as rapid prototyping (Huang and Lin 2003), or involving complex robot systems, such as cooperating robots (Montemayor and Wen 2005), cooperating teleoperation systems (Sirouspour 2005) or humanoid robots (Sugihara and Nakamura 2002), cannot be programmed using the geometric primitives present in first-generation robot controllers. Furthermore, even for tasks which actually can be programmed using these primitives, the process of hand-coding all paths is a burdensome task for all but the most simple applications. This, combined with the desire to simulate complete robot environments with possibly multiple robots and extensive peripheral equipment, inspired the development of CAD-based systems to simulate and control robots, such as Delmia V5 Robotics (Delmia 2006). Using such CAD-based software any robot motion can be checked for collisions, reachability, accuracy of the motion, etc.

While this way of programming robots is a huge step forward with respect to initial robot programming support, research focused on a multitude of topics which are not covered by current commercially available software. For instance, there is little to no support for the inclusion of sensor data to alter the task execution online. Also in the field of kinematic modeling and path resolution, a lot of research progress has been made compared to what is currently commercially available. An active research topic is the exploitation of task and robot redundancy to achieve better performance. Owen, Croft, and Benhabib (2005) present a weighted pseudo-inverse method with a weighting matrix based on joint torque, to reduce the acceleration for joints which are near torque saturation. When applied to offline planning, trajectories that would normally be infeasible due to torques exceeding their limits, can be made feasible as their method brings the saturated torques below the torque limits. In (Wu, Cui, and Chen 2000) and (Ahmad and Luo 1989), redundancy is used to realize a task while avoiding singularities and joint limits, while in (Jouaneh, Wang, and Dornfeld 1990; Jouaneh, Dornfeld, and Tomizuka 1990) optimal paths are resolved based on minimum energy and time criteria. English and Maciejewski (2000) identify that many of these velocity control techniques can be cast into a particular representation of the Liégois method (Liegeois 1977; Aksenov, Voronetskaya, and Fomin 1978). In this method, redundancy is resolved by specifying a set of constraints (the *secondary* constraints) which are realized in the null-space of the task constraints (the *primary constraints*).

Complex Task Dynamics

Also with respect to modeling task and robot dynamics, a lot of progress has been made in research which is not yet reflected in commercially available robot controllers or control software. Currently, much focus goes to simulation and control of humanoid robots. Chang and Khatib (2000) present efficient algorithms to model and control branching mechanisms, such as humanoid robots. Combined with collision and contact models (Ruspini and Khatib 1999), these algorithms are used for interactive simulation of humanoid robots (Khatib et al. 2003). In (Khatib et al. 2004), a framework for whole-body control of humanoid robots is presented, which decouples the interaction between the task and postural objectives. To ensure that the secondary (postural) objectives do not interfere with the primary (task) objectives, the principles of (Nakamura, Hanafusa, and Yoshikawa 1987) are applied on a dynamic level, with the operational space inertia matrix as the natural weighting matrix for the projection used in solving the redundancy problem. In (Bruyninckx and Khatib 2000) Gauss' Principle of Least Constraint is used to derive these 'natural' dynamic equations for redundant manipulators.

Other research focused on the combination of dynamic systems with kinematic constraints. (Bonaventura and Jablokow 2005) present a modular approach for the dynamic modeling and simulation of complex robot systems, composed of multiple robots constrained by multiple concurrent contacts (this is, kinematic constraints). In (Liu, Xu, and Bergerman 1999) modeling and control of multiple cooperative underactuated manipulators, handling a rigid object is discussed.

Task Level Programming

A completely different approach to task specification, though primarily as a research topic, is that of task level programming. In task level programming, the operator specifies *what* the goals are for a certain task, but not *how* these goals should be accomplished. It is up to the underlying planning system to come up with a strategy to realize the specification, and to generate motion commands for the robot. Seminal work in the field of mechanical assembly has been done by Ambler and Popplestone (1975). Ambler and Popplestone specify geometric relations (that is, geometric constraints) between features of two objects (Popplestone, Weiss, and Liu 1988). The goal is to infer the desired relative *pose* of these objects from the specified geometric relations between the features. In turn this desired relative pose is used to determine the goal pose of a robot who has to assemble both objects. The geometric relations are based on pose closure equations and they are solved using symbolic methods. To support the programmer with the specification, feature frames and object frames are introduced, as well as suitable local coordinates to express the relative pose between these frames. In a similar way, motion planning methods in configuration space (C-space) (Lozano-Pérez 1983) specify relative pose as the result of applying constraints to objects. Robot path planning is reduced to path planning of a point in C-space, complying to the constraints (Chen and Zelinsky 2003). In (Latombe 1991; Latombe 1999; LaValle 2006), a survey is given of research in planning algorithms. While task level programming has been an active research topic since long (Taylor 1976; Latombe 1989), it is still not mature enough to allow for widespread application.

2.3.2 Sensor-based tasks

Sensor feedback is introduced in robotics to overcome model inconsistencies, that is, differences between what is modeled and what is present in the real robot environment. The sensor measurements are used in the control loop, to adapt the task execution to these variations. The most common sensors in robotics are force/torque sensors and cameras for manipulation tasks, and cameras, laser range finders and ultrasonic sensors in mobile robotics. This section gives an overview of sensor-based robotics, with main focus on force control (Siciliano and Villani 1999; Canudas de Wit, Siciliano, and Bastin 1996) and compliant motion tasks (De Schutter and Van Brussel 1988a), as these topics are historically the main research topics in sensor-based manipulation at PMA, K.U.Leuven.

Compliant Motion

Compliant motion task are tasks in which objects are moved in contact, such as assembly, deburring or polishing. A force/torque sensor measures the contact wrench, and motion is controlled such, that the contact wrench is regulated to a desired value. The three main approaches to force control are *hybrid force/position control* (hereafter called Hybrid control), *impedance control* and *parallel force/position control* (hereafter called Parallel control). Hybrid control (Raibert and Craig 1981; Fisher and Mujtaba 1992) is based on a decomposition of the workspace according to the configuration of the contacting objects, into n purely motion controlled directions and $6 - n$ purely force controlled directions. This separation into motion and force controlled subspaces is called the *Hybrid Control Paradigm* (HCP). In impedance control (Hogan 1985; Hogan 1987), position and force are controlled in such a way that a desired mechanical impedance of the end-effector, to external forces exerted by contact with the environment, is obtained. It is a generalization of stiffness control (Salisbury 1980) and admittance control (Whitney 1977).

Parallel control was proposed in (Chiaverini and Sciavicco 1993) and (Siciliano 1995), and in (De Schutter and Van Brussel 1988c) and (De Schutter 1988), where it was termed *feedforward motion in a force controlled direction*. It is an extension to the HCP, combining force and motion control in a single direction. The force control dominates the motion control in each direction: in case of conflict, the force setpoint is regulated at the expense of a position error. In (Anderson and Spong 1988), hybrid impedance control is introduced as a combination of Hybrid control and Impedance control.

Lipkin and Duffy (1988), Duffy (1990) and Doty, Melchiorri, and Bonivento (1993) have pointed out the importance of invariant descriptions when dealing with twists and wrenches to represent 3D velocity of a rigid body and 3D force interaction, respectively. Non-invariant descriptions result in a different robot behavior when changing the reference frame in which twists and wrenches are expressed, when changing the reference point for expressing the translational velocity or the moment, or when changing units. Such non-invariance is highly undesirable in practical applications as it compromises the predictability of the resulting robot behavior. Several ways to deal with this problem have been proposed, for instance in (Lipkin and Duffy 1988; Doty, Melchiorri, and Bonivento 1993; Jankowski and ElMaraghy 1996; De Schutter, Torfs, Dutré, and Bruyninckx 1997). These approaches have in common that a metric is introduced, which defines a physically relevant norm of the wrenches and twists, and that a weighted pseudo-inverse is used accordingly.

The original formulation of the HCP neglects the dynamics of the manipulator, which can cause unstable response (An and Hollerbach 1989). This led to more precise formulations of the HCP, taking the arm dynamics into account. Khatib (1987) developed the operational space formulation and Yoshikawa, Sugie, and Tanaka (1988) introduced the dynamic hybrid control approach.

Several review papers on force control have been published (Yoshikawa 2000; De Schutter and Bruyninckx 1996; Whitney 1987; Natale 2003), of which (De Schutter et al. 1997) and (Caccavale et al. 2005) are of particular interest. The former paper takes a look at force control from a distance to put force control into a broader context, and the latter focuses on embedding force control into industrial robots. Related work was presented in (Ferretti, Magnani, and Rocco 2004), in which an impedance controller is proposed for industrial manipulators with specific attention to the industrial aspects of the manipulator (decentralized PID position control and torsional flexibility and friction of the joints), and in (Caccavale et al. 2005), which focuses on impedance and parallel control with respect to implementation on industrial robots. While force control has been a research topic since the 1970's, the implementation of force control on an off-the-shelf industrial manipulator is still not straightforward, mostly due to the closedness of industrial controllers with respect to interfacing the control loop to integrate sensor measurements (Miller and Lennox 1990; Van de Poel, De Schutter, and Van Brussel 1994). Recent controllers, such as the Stäubli CS8C (Stäubli 2006; Garćia et al. 2006) or the ABB IRC5 (ABB 2006; Garćia et al. 2005; Mustapic et al. 2004) provide more open architectures and are promising with respect to sensor-based robotics.

Task specification for compliant motion tasks

Several task specification approaches for force controlled tasks have been developed. The *Compliance Frame* (Mason 1981) or *Task Frame Formalism* (TFF) (De Schutter and Leysen 1987) is such a task specification support formalism for tasks according to the HCP, in which the force and velocity controlled directions coincide with the axes of a frame (the *Task Frame*). An extensive catalog of TFF models and specifications can be found in (Bruyninckx and De Schutter 1996). COMRADE (Compliant Motion Research and Development Environment) is a compliant motion specification language based on the TFF (Van de Poel et al. 1993; Witvrouw et al. 1995; Witvrouw 1996). It is an implementation of the general scheme: it provides the controller, generator,..., needed to execute a task according to the TFF. Task specification using COMRADE is reduced to the initialization of these different components. The user has to initialize:

1. the model, by specifying the Task Frame and selecting the force and velocity controlled directions,
2. the generator, by specifying the desired values for each force and velocity,
3. the DES, by specifying the termination conditions for each force controlled motion.[1]

COMRADE provides a set of standard controllers, such as velocity or force controllers[2]. According to the model, the correct controllers for the task are selected. The measured force and joint positions are the inputs to the control program. The outputs are the joint velocities. For tasks with a varying position of the Task Frame, COMRADE provides two estimators to track the Task Frame: *track-on-force* and *track-on-velocity* (Witvrouw 1996). The DES monitors the termination conditions and switches states accordingly.

In (Wang 1999), a human demonstration approach was presented to specify tasks according to the TFF. Other extensions of the TFF have been presented

[1]For instance, if a force, substantially larger than the friction force, is felt in a velocity controlled direction, a transition has occurred to a different contact formation.

[2]Of course, different controllers can be implemented, and the parameters of those controllers can be altered by the user.

in (Baeten and De Schutter 2003; Baeten, Bruyninckx, and De Schutter 2003), in which vision and force are combined for robotic servoing using the TFF, and in (Kröger, Finkemeyer, and Wahl 2004), in which the Task-Net is introduced. The Task-Net is an extensive implementation of the TFF, with emphasis on the DES component of the control program. It offers a programming language to define a state diagram, in which state transitions are triggered by sensor measurements (Kröger et al. 2004).

The TFF is conceptually simple, and can be applied to a multitude of contact tasks. However, it cannot model every contact formation, for instance with multiple point contacts at different faces (Bruyninckx and De Schutter 1997). For these applications, the environment poses geometric constraints on the motion of the contacting object, which cannot be modeled by the Task Frame. Several approaches have been presented which go further than the TFF in this respect, and explicitly take a more elaborate geometric model of the constraints into account. Usually, these are based on the dynamical formulation of the HCP (Khatib 1987). In (Featherstone, Sonck, and Khatib 1999) a first-order kinematic model of the rigid-body contact is considered. (Liu and Li 2002) is a generalization of this work, based on work by Blajer (1997), who proposed a geometric treatment of constrained mechanical systems. Their approach considers general constraint systems, and defines two orthogonal subspaces: the subspace of constraint forces (this is the wrench space) and the tangent space of the constraint manifold for holonomic constraints (this is the twist space complying to the geometric constraints). The system dynamics are then projected into two orthogonal components, for which geometrical interpretation is given. Based on these projections, a hybrid position/force control algorithm is proposed. A similar approach is followed in (Featherstone 2004), which deals with contacts between bodies which are subject to additional kinematic constraints from some other source, and in (Aghili 2005), in which a control scheme is proposed based on projection of the inverse dynamics, minimizing the weighted Euclidean norm of the actuation force.

Constraint-based programming

More general than the TFF or the formalisms based on the definition of wrench and twist subspaces according to the contact configuration, is to assign control modes and corresponding constraints to *arbitrary* directions in the six dimensional manipulation space. This approach, known as *constraint-based programming*, opens up new applications involving a much more complex geometry (for example compliant motion with two- or three-point contacts) and/or involving multiple sensors that control different directions in space simultaneously (De Schutter et al. 2005). In a sense, the Task Frame is replaced by and extended to multiple *feature frames*.

Samson and coworkers (Samson, Le Borgne, and Espiau 1991) have put a major step forward in the area of constraint-based programming. They introduce the *task function approach*. The basic idea behind the approach is that many robot tasks may be reduced to a problem of positioning, and that the control problem boils down to regulating a vector function, known as the task function, which characterizes the task. The variables of this function are the joint positions and time. Their work already contains the most important and relevant aspects of constraint-based programming. Based on the task function they propose a general non-linear proportional and derivative control scheme of which the main robotic control schemes are special cases. In addition they analyze the stability and the robustness of this control scheme. They also consider task functions involving feedback from exteroceptive sensors, they recognize the analogy with the kinematics of contact and they treat redundancy in the task specification as a problem of constrained minimization. Finally, they derive the task function for a variety of example applications using a systematic approach. Espiau and coworkers (Espiau, Chaumette, and Rives 1992) apply this approach to visual servoing and show how this task can be extended to a hybrid task consisting of visual servoing in combination with, for example, trajectory tracking. Very recent work by Samson (Fruchard, Morin, and Samson 2006) presents a framework for systematic and integrated control of *mobile manipulation*, for both holonomic and non-holonomic robots; the scope of that framework is much more focused on only feedback control, while this thesis also integrates the instantaneous specification and the estimation of sensor-based tasks.

In addition to constraints on *position* level, constraints on velocity and acceleration level have traditionally been coped with, for example in mobile robotics and multibody simulation. These constraints are expressed using velocity and acceleration closure equations, and they are solved for the robot position, velocity or acceleration using numerical methods that are standard in multibody dynamics. Basically, these numerical methods are of either the *reduction* type (identifying and eliminating the dependent coordinates, see (Critchley and Anderson 2003) for an overview and further reference) or the *projection* type (doing model updates in a non-minimal set of coordinates and then projecting the result on the allowable motion manifold, see (de Jalón and Bayo 1993) for an overview). The latter approach is better known in robotics as *(weighted) pseudo-inverse control* of redundant robots, (Doty, Melchiorri, and Bonivento 1993; Klein and Huang 1983; Nakamura 1991).

2.4 Estimation

Sensor feedback is introduced in robotics to overcome model inconsistencies. As the structure of the model is usually straightforward to define, these inconsistencies are mostly due to incorrectly assessed parameters of the model.

For instance, a table top is often modeled accurately enough by a plane, but it is not necessarily straightforward to accurately identify the actual height of the table with respect to the robot base. While these model inconsistencies can be overcome to some extent by adapting the task execution to the sensor measurements, they also provide information about the parameters of the model, and estimation techniques can be used to adapt the model accordingly.

Most approaches to the estimation of model parameters only use instantaneous sensor measurements, and apply ad-hoc, non-stochastic algorithms. More recent work is based on stochastic methods, and takes uncertainty on the sensor measurements into account. The uncertainty consists of sensor noise and modeling uncertainty such as for instance inaccuracy of the wrench measurements due to contact friction. A typical example of such a stochastic method is the Kalman filter (Kalman 1960; Sorenson 1985; Sorenson 1970), and its variants: the extended (EKF) and iterated extended Kalman filter (IEKF) (Tanizaki 1996), and, more recently, the non-minimal state Kalman filter (NMSKF) (Lefebvre, Bruyninckx, and De Schutter 2005b). Particle filters (sequential Monte Carlo methods) (Doucet, Gordon, and Krishnamurthy 2001) are computationally more expensive estimation methods, but can cope with highly non-linear and multi-modal estimation problems, possibly with hybrid states, that is, consisting of continuous and discrete variables.

The rest of this section briefly discusses applications of estimation in robotics. Three topics are covered: the estimation of *geometric parameters*, the estimation of *dynamic parameters*, and, less common but gaining importance as research focuses on human-robot cooperation tasks, the estimation of *human intention*.

2.4.1 Estimation of geometric parameters

As many robot tasks are purely geometric, the estimation of geometric parameters is of particular importance in robotics. Most approaches use only instantaneous sensor measurements and apply ad-hoc, non-stochastic algorithms. For instance, Mason and Salisbury (1985) estimate the contact situation from force measurements, by calculating the line of force that corresponds to the measured force and torque, and De Schutter and Van Brussel (1988b) use a deterministic velocity-based observer to determine the contact geometry.

The Kalman filter (Kalman 1960) is a well accepted recursive estimation technique in sensor-based robotics in general, especially in mobile robotics (Leonard and Durrant-Whyte 1992) and computer vision (Soatto, Frezza, and Perona 1996). More recently, the gain in computer processing power led to a wider acceptance of sequential Monte Carlo or particle filter approaches (Doucet, Gordon, and Krishnamurthy 2001), which can handle large uncertainties, even with highly non-linear systems.

In mobile robotics, simultaneous localization and mapping (SLAM) is one of the main geometrical estimation problems, dealing with the identification of the position of one or more robots with respect to a map, which is simultaneously being built up (Leonard and Durrant-Whyte 1991). Recently, hierarchical methods are presented in which a large environmental map is built up from independent local stochastic maps (Estrada, Neira, and Tardós 2005). Similar problems are present in computer vision, for instance to estimate the pose of a camera while concurrently tracking image features (Davison 2003).

Somewhat later than in mobile robotics, Kalman filters were also introduced for manipulation and compliant motion tasks. In (De Schutter et al. 1999), a general contact model and a measurement equation based on reciprocity is presented to estimate the first-order geometric parameters of a point face contact (this is, the orientation of contact normals and location of contact points) and their time-variance that occur in force controlled compliant motions, and to monitor transitions between contact situations. Cortesão, Park, and Khatib (2003a) present an adaptive approach to haptic manipulation using Kalman observers to estimate the contact force online. In (Cortesão, Park, and Khatib 2003b), an extension to this approach is presented to also estimate the contact stiffness, while in (Park and Khatib 2005) their approach is extended to multiple contacts, possibly on different links of the robot. In (Slaets et al. 2007), a 3D geometrical model is constructed using a non-minimal state Kalman filter (NMSKF) (Lefebvre, Bruyninckx, and De Schutter 2005a). Sminchisescu, Metaxas, and Dickinson (2005) cover a similar problem in computer vision: adaptive object shape estimation and tracking in camera images.

2.4.2 Estimation of dynamical parameters

The dynamical parameters of the robot and its load are usually difficult to measure or obtain, especially for robots which are not developed in-house, such as industrial robots. As these parameters are needed for control approaches which explicitly take into account the dynamics of the robot, such as the operational space formalism (Khatib 1987), they are often the subject of estimation.

The different approaches to the estimation of dynamical parameters can be divided into two categories. In the most widely used identification approach, the parameters are estimated from 'internal' data of the robot (Gautier 1986; Swevers et al. 1996). Motion data is obtained from the built-in encoders of the robot, and actuator data is obtained in the form of actuator current measurements. The dynamical model resulting from these measurements is called the *internal model*. In the other approach, an external sensor is used to measure the reaction forces and torques at the base plate of the robot. The resulting dynamical model is called the *reaction* or *external model* (Liu, Dubowsky, and Morel 1998). In (Verdonck, Swevers, and Samin 2001), the internal and external measurements are combined into one model.

2.4.3 Estimation of human intent

A complete other research area involving estimation is that of assessing the intention of the human operator during task execution, for instance in human-robot cooperation or in teleoperation tasks. An interesting example is that of (Glover, Thrun, and Matthews 2004), in which hierarchical semi-Markov Models are used to estimate human intent based on topological data and time-of-day distributions. Numerous other approaches and applications have been presented, such as fuzzy inference on physiological signals (Kulić and Croft 2003), Bayesian techniques (Demeester et al. 2003), or spectral pattern recognition in force signals (Fernandez et al. 2001). Bruemmer et al. (2005) go a step further, in that they study representations which allow the human and robot to predict behavior and communicate intent in human-robot collaboration tasks. Bredereke and Lankenau (2005) discuss the problem of mode confusions in shared-control systems, which occur when the observed behavior of the system is out of sync with the user's mental model of its behavior.

2.5 Conclusion

This chapter gives an overview of the literature on task specification for and estimation in robot tasks. For a limited set of sensor-based tasks, such as those which comply to the TFF, extensive programming support is available. For more complex tasks however, for instance using redundant robotic systems such as mobile manipulator arms, cooperating robots, robotic hands or humanoid robots, or using multiple sensors such as vision, force/torque and distance sensors, little to no programming support exists. As a result, the task programmer has to rely on extensive knowledge in multiple fields such as spatial kinematics, 3D modeling of objects, geometric uncertainty and sensor systems, dynamics and control, estimation, as well as resolution of redundancy and of conflicting constraints.

The goal of our research is to fill this gap. We want to develop programming support for the implementation of complex, sensor-based robotic tasks in the presence of geometric uncertainty. The foundation for this programming support, presented in this thesis, is iTASC (instantaneous Task Specification using Constraints): a generic and systematic approach to specify and control a task while dealing properly with geometric uncertainty. Inspired by Ambler and Popplestone (1975), feature frames and object frames are introduced. A set of auxiliary coordinates, denoted as feature twist coordinates, is introduced to model the motion of the object and feature frames. By specifying constraints on the feature twist coordinates, the first order expression of the Task Function (Samson, Le Borgne, and Espiau 1991) is obtained, as needed for velocity resolved robots. The approach considers both redundant and overconstrained task specifications, and uses well-known approaches to deal with these situations (Doty, Melchiorri, and Bonivento 1993; Nakamura 1991).

Whereas most previous work relies on the intrinsic robustness of sensor based control to deal with geometric uncertainty, in iTASC geometric uncertainty is explicitly modeled. To this end an additional set of uncertainty coordinates is introduced. These coordinates are then estimated using techniques such as the Kalman filter (Kalman 1960).

Chapter 3

Mathematical Preliminaries

This chapter introduces some mathematical concepts and notation, which are assumed known in the further chapters of this thesis.

3.1 Rigid Body Kinematics

3.1.1 Rigid Body Pose

The *pose* of an object with respect to some reference is its relative position and orientation. At least six parameters are needed to fully describe an object's pose. Different choices for these parameters are possible, such as Cartesian or spherical coordinates to describe the position, and Euler or Roll-Pitch-Yaw angles for the orientation.

A *finite displacement* $\boldsymbol{d}_a^b$ is a six-vector expressing the pose of object b with respect to object a, or rather, the relative pose of reference frames a and b on these objects:

$$\boldsymbol{d}_a^b = \left[\begin{array}{cccccc} x & y & z & \phi & \theta & \psi \end{array} \right]^T .$$

In this, (x, y, z) is a representation for the position and (ϕ, θ, ψ) a representation for the orientation. Without loss of generality, this text assumes Cartesian coordinates for the position and $ZYX-$Euler angles for the orientation, unless stated otherwise. In general, the composition of finite displacements is not commutative, as no set of three angles to represent an orientation is a vector: contrary to vector addition, the addition of two sets of Euler angles (for instance) does not give the set of Euler angles that corresponds to the composed orientation. Furthermore, the order of rotations matters: the composition of rotations is not commutative.

A set of six coordinates to describe a pose is said to be *minimal.* Often non-minimal representations, that is, with more than six parameters, are used to describe a pose. Such non-minimal representations have improved properties with respect to numerical stability and unambiguity in the representation (as every orientation representation with only three parameters inevitably has coordinate singularities at a number of orientations), but require extra computational cost because of the need to carry along a number of constraints between the numbers in the non-minimal representation.

A common non-minimal representation of the relative pose of a frame b with respect to a frame a is the *homogeneous transformation matrix* $\boldsymbol{T}_a^b$:

$$\boldsymbol{T}_a^b = \begin{bmatrix} \boldsymbol{R}_a^b & {}_a\boldsymbol{p}^{a,b} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix},$$

where ${}_a\boldsymbol{p}^{a,b}$ represents the position vector from the origin of a to the origin of b, expressed in frame a, and $\boldsymbol{R}_a^b$ represents the orientation matrix of b with respect to a. The homogeneous transformation of a composition of relative poses is found by multiplying the individual transformation matrices:

$$\boldsymbol{T}_a^n = \boldsymbol{T}_a^b\, \boldsymbol{T}_b^c \quad \ldots \quad \boldsymbol{T}_m^n.$$

3.1.2 Rigid Body Motion

The *twist* $\mathbf{t}$ of an object is a six-vector $\left[\begin{smallmatrix}\boldsymbol{v}\\ \boldsymbol{\omega}\end{smallmatrix}\right]$, containing the object's translational and rotational velocities $\boldsymbol{v}$ and $\boldsymbol{\omega}$ (both 3×1 vectors).

At each time instant, an object has a unique translational and rotational velocity (possibly zero). Hence, the twist is unique and unambiguously defined. In other words, it is a kinematic property of the object. However, the mathematical representation of a twist (that is, the numerical values for the coordinates of $\boldsymbol{v}$ and $\boldsymbol{\omega}$) depends on the choice of a *reference point* and *reference frame*. Because of this, leading sub- and superscripts are added to the notation of a twist: ${}^c_d\mathbf{t}_a^b$ expresses the twist of b with respect to a, with reference point c and reference frame d. These sub- and superscripts are omitted when it is clear from the context which twist is considered.

Figure 3.1 shows an example. An object rotates around an axis, which is oriented along the Z-axis of a reference frame f. Two points are defined on the object: point a resides on the rotation axis, contrary to point b. Both points are fixed to the object. As point a resides on the rotation axis, the twist of the object, expressed in a, only has a rotational component:

$${}^a_f\mathbf{t} = \begin{bmatrix} \boldsymbol{0}_{3\times 1} \\ {}_f\boldsymbol{\omega} \end{bmatrix},$$

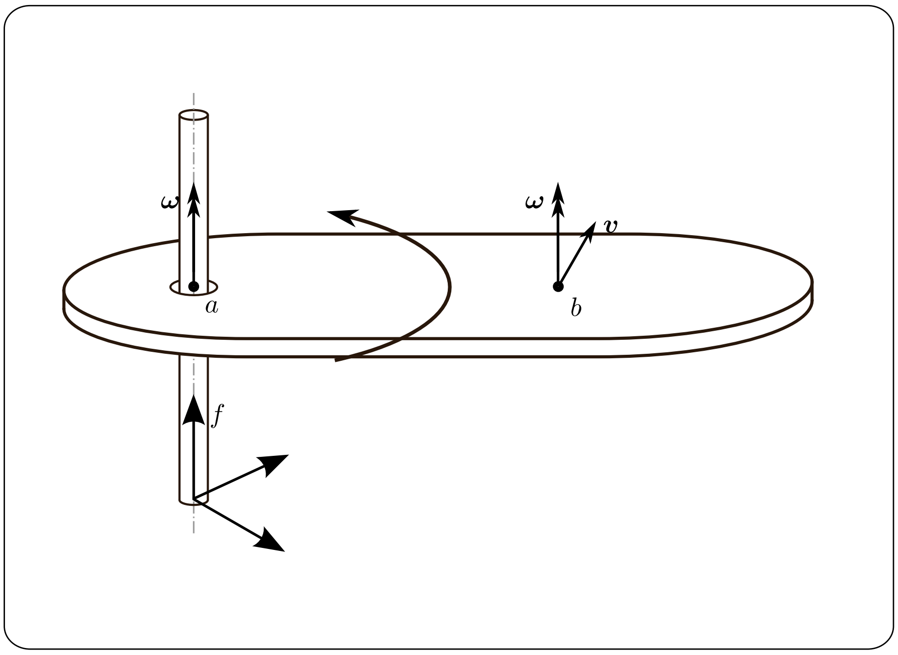


FIGURE 3.1: An object rotates around a vertical axis. Point a of this object, on the axis, purely rotates. Point b of this object, not on the axis, translates as well as rotates.

with ${}_f\boldsymbol{\omega}$ the rotational velocity vector of the object: ${}_f\boldsymbol{\omega} = \left[\, 0 \; 0 \; \omega \,\right]^T$.
As point b does not reside on the rotation axis, the twist of the object, expressed in b, has both a translational and rotational component:

$$ {}^b_f\mathbf{t} = \begin{bmatrix} {}_f\boldsymbol{v} \\ {}_f\boldsymbol{\omega} \end{bmatrix}. $$

In this, ${}_f\boldsymbol{v} = {}_f\boldsymbol{p}^{a,b} \times {}_f\boldsymbol{\omega}$, with ${}_f\boldsymbol{p}^{a,b}$ the vector from a to b, expressed in f. Both ${}^a_f\mathbf{t}$ and ${}^b_f\mathbf{t}$ express the same motion of the object and as such *the* twist of the object. However, the values of their coordinates differ as different reference points are considered. Similarly, a change in reference frame changes the coordinates of $\boldsymbol{v}$ and $\boldsymbol{\omega}$.

The twist of a composition of relative motions is found by adding the individual twists:

$$ \mathbf{t}_a^n = \mathbf{t}_a^b + \mathbf{t}_b^c + \cdots + \mathbf{t}_m^n. \tag{3.1} $$

In contrary to the addition of finite displacements, the addition of twists is commutative, as twists are vectors. Equation (3.1) is valid irrespective of the choice of reference points and frames for each of the twist. However, to mathematically evaluate (3.1), all twists should be referred to the same reference point and frame. The 6×6 *screw projection matrix* ${}^{f}_{g}\boldsymbol{P}$ transforms the coordinate representation of a twist ${}^{a}_{f}\mathbf{t}$ from its current reference frame f to a new reference frame g, without changing the reference point:

$$ {}^{a}_{g}\mathbf{t} = {}^{f}_{g}\boldsymbol{P}\ {}^{a}_{f}\mathbf{t}, \text{ with } {}^{f}_{g}\boldsymbol{P} = \begin{bmatrix} \boldsymbol{R}^{f}_{g} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{R}^{f}_{g} \end{bmatrix}. $$

The 6×6 *reference point transformation matrix* $\boldsymbol{M}^{a}_{b}$ transforms the coordinate representation of a twist ${}^{a}_{f}\mathbf{t}$ from its current reference point a to a new reference point b, without changing the reference frame:

$$ {}^{b}_{f}\mathbf{t} = \boldsymbol{M}^{a}_{b}\ {}^{a}_{f}\mathbf{t}, \text{ with } \boldsymbol{M}^{a}_{b} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & [{}_{f}\boldsymbol{p}^{b,a}\times] \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{bmatrix}. $$

In this, $[\boldsymbol{p}\times]$ is the 3×3 skew-symmetric matrix representing the cross product with a 3×1 vector $\boldsymbol{p}$:

$$ [\boldsymbol{p}\times] = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix}. \tag{3.2} $$

The time derivative of a finite displacement $\dot{\boldsymbol{d}}$ is another way to describe an object's motion. The relation between this time derivative and the corresponding twist is given by:

$$ \begin{aligned} \dot{\boldsymbol{d}}^{b}_{a} &= \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{E}^{-1} \end{bmatrix} {}^{b}_{a}\mathbf{t}^{b}_{a}, \\ &\equiv \boldsymbol{\mathcal{E}}^{b}_{a}\mathbf{t}^{b}_{a}, \end{aligned} $$

in which $\boldsymbol{E}$ is the *integrating factor*: a 3×3 matrix which converts the time rates $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ of the chosen representation for the orientation, to an angular velocity $\boldsymbol{\omega}$. For $ZYX-$Euler angles, $\boldsymbol{E}$ is given by:

$$ \boldsymbol{E} = \begin{bmatrix} 0 & -\sin\phi & \cos\theta\cos\phi \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 1 & 0 & -\sin\phi \end{bmatrix}, $$

and

$$ \boldsymbol{E}^{-1} = \begin{bmatrix} \dfrac{\cos\phi\sin\theta}{\cos\theta} & \dfrac{\sin\phi\sin\theta}{\cos\theta} & 1 \\ -\sin\phi & \cos\phi & 0 \\ \dfrac{\cos\phi}{\cos\theta} & \dfrac{\sin\phi}{\cos\theta} & 0 \end{bmatrix}. $$

Note that this inverse relation becomes singular for $\theta = \pi$ or $-\pi$. A possible solution is to choose another angular representation in the neighborhood of these singularities.

3.1.3 Exponential and Logarithm

In this section, the relation between a twist and the rate of the transformation matrix is derived. First, the case of a pure rotation is considered. Then, a similar reasoning is made for a twist with both a translational and a rotational component.

Exponential of a rotational velocity

Consider two frames a and b, with initial pose:

$$\boldsymbol{T}_a^b = \begin{bmatrix} \boldsymbol{R}_{a,init}^b & 0 \\ \boldsymbol{0} & 1 \end{bmatrix}. \tag{3.3}$$

Frame b rotates with respect to frame a, with a constant rotational velocity ${}_a\boldsymbol{\omega}$. The coordinates in a, of a point p fixed to b, are:

$${}_a\boldsymbol{p} = \boldsymbol{R}_a^b \; {}_b\boldsymbol{p}, \tag{3.4}$$

with ${}_b\boldsymbol{p}$ constant. The time derivative of (3.4) gives the instantaneous translational velocity of the point p:

$$\begin{aligned} {}_a\dot{\boldsymbol{p}} &= \dot{\boldsymbol{R}}_a^b \; {}_b\boldsymbol{p}, & (3.5)\\ &= \dot{\boldsymbol{R}}_a^b \boldsymbol{R}_a^{bT} \; {}_a\boldsymbol{p}. & (3.6) \end{aligned}$$

Alternatively, the translational velocity of p is given by:

$${}_a\dot{\boldsymbol{p}} = [{}_a\boldsymbol{\omega}\times] \, {}_a\boldsymbol{p}, \tag{3.7}$$

in which $[{}_a\boldsymbol{\omega}\times]$ denotes the matrix form of the cross product with ${}_a\boldsymbol{\omega}$, as defined in (3.2). Hence, from (3.6) and (3.7):

$$\dot{\boldsymbol{R}}_a^b \boldsymbol{R}_a^{bT} = [{}_a\boldsymbol{\omega}\times], \tag{3.8}$$

or:

$$\dot{\boldsymbol{R}}_a^b = [{}_a\boldsymbol{\omega}\times] \, \boldsymbol{R}_a^b. \tag{3.9}$$

The solution to this equation relates the rotation matrix at a time t to the initial rotation matrix, for a constant rotational velocity ${}_a\boldsymbol{\omega}$:

$$\boldsymbol{R}_a^b(t) = \exp\left([{}_a\boldsymbol{\omega}\times] \, t\right) \boldsymbol{R}_{a,init}^b, \tag{3.10}$$

with exp the matrix exponential.

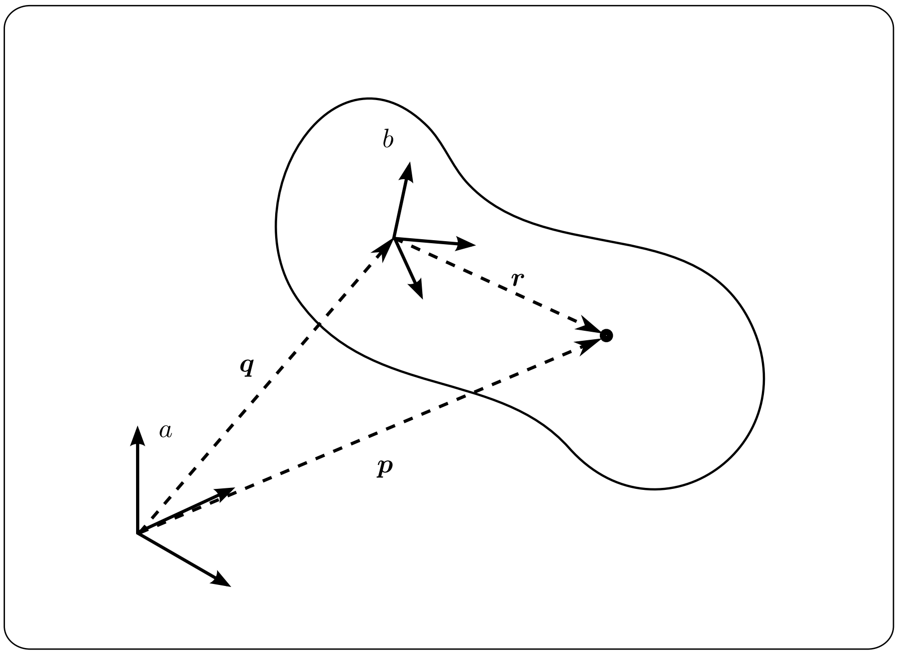


Figure 3.2: An object moves with respect to a frame a. Frame b is attached to the object.

Exponential of a twist

A similar reasoning can be made for motion consisting of both translation and rotation. Figure 3.2 shows a body which moves with respect to a frame a. Frame b is attached to the body. Consider a point of the object, with constant coordinates ${}_b\boldsymbol{r}$ with respect to b. Its coordinates with respect to a are given by:

$$ {}_a\boldsymbol{p} = \boldsymbol{R}_a^b \; {}_b\boldsymbol{r} + {}_a\boldsymbol{q}. \tag{3.11}$$

The time derivative of (3.11) gives the translational velocity of the point:

$$\begin{aligned} {}_a\dot{\boldsymbol{p}} &= \dot{\boldsymbol{R}}_a^b \; {}_b\boldsymbol{r} + {}_a\dot{\boldsymbol{q}}, && (3.12)\\ &= \dot{\boldsymbol{R}}_a^b \left(\boldsymbol{R}_a^{b\,T} ({}_a\boldsymbol{p} - {}_a\boldsymbol{q}) \right) + {}_a\dot{\boldsymbol{q}}, && (3.13)\\ &= \dot{\boldsymbol{R}}_a^b \boldsymbol{R}_a^{b\,T} {}_a\boldsymbol{p} - \dot{\boldsymbol{R}}_a^b \boldsymbol{R}_a^{b\,T} {}_a\boldsymbol{q} + {}_a\dot{\boldsymbol{q}}. && (3.14) \end{aligned}$$

In matrix notation, this can be written as:

$$\begin{bmatrix} {}_a\dot{\boldsymbol{p}} \\ 0 \end{bmatrix} = \dot{\boldsymbol{T}}_a^b \; \left(\boldsymbol{T}_a^b \right)^{-1} \begin{bmatrix} {}_a\boldsymbol{p} \\ 1 \end{bmatrix}, \tag{3.15}$$

with

$$\dot{\boldsymbol{T}}_a^b = \begin{bmatrix} \dot{\boldsymbol{R}}_a^b & {}_a\dot{\boldsymbol{q}} \\ \boldsymbol{0} & 0 \end{bmatrix}. \tag{3.16}$$

Alternatively, with ${}_a\boldsymbol{\omega}$ the rotational velocity of the object, the velocity of the point can be expressed as:

$$\begin{aligned} {}_a\dot{\boldsymbol{p}} &= {}_a\dot{\boldsymbol{q}} + [{}_a\boldsymbol{\omega}\times]\ {}_a\boldsymbol{r}, && (3.17)\\ &= {}_a\dot{\boldsymbol{q}} + [{}_a\boldsymbol{\omega}\times]\left({}_a\boldsymbol{p} - {}_a\boldsymbol{q}\right), && (3.18)\\ &= [{}_a\boldsymbol{\omega}\times]\, {}_a\boldsymbol{p} + {}_a\boldsymbol{v}, && (3.19) \end{aligned}$$

with:

$${}_a\boldsymbol{v} = {}_a\dot{\boldsymbol{q}} - [{}_a\boldsymbol{\omega}\times]\, {}_a\boldsymbol{q}. \tag{3.20}$$

${}_a\boldsymbol{v}$ corresponds to the velocity of that –possibly imaginary– point of the object, which instantaneously coincides with the origin of a. In matrix notation, (3.19) is written as:

$$\begin{bmatrix} {}_a\dot{\boldsymbol{p}} \\ 0 \end{bmatrix} = \begin{bmatrix} [{}_a\boldsymbol{\omega}\times] & {}_a\boldsymbol{v} \\ \boldsymbol{0} & 0 \end{bmatrix} \begin{bmatrix} {}_a\boldsymbol{p} \\ 1 \end{bmatrix}. \tag{3.21}$$

Hence, from (3.15) and (3.21):

$$\dot{\boldsymbol{T}}_a^b \ \left(\boldsymbol{T}_a^b\right)^{-1} = \begin{bmatrix} [{}_a\boldsymbol{\omega}\times] & {}_a\boldsymbol{v} \\ \boldsymbol{0} & 0 \end{bmatrix}, \tag{3.22}$$

or:

$$\dot{\boldsymbol{T}}_a^b = \begin{bmatrix} [{}_a\boldsymbol{\omega}\times] & {}_a\boldsymbol{v} \\ \boldsymbol{0} & 0 \end{bmatrix} \boldsymbol{T}_a^b\,. \tag{3.23}$$

The solution to this equation relates the pose of the object at a time t to the initial pose, for a constant twist of the object $\mathbf{t}_a^{object} = \left[\begin{smallmatrix} {}_a\boldsymbol{v} \\ {}_a\boldsymbol{\omega} \end{smallmatrix}\right]$:

$$\boldsymbol{T}_a^b(t) = \exp\left(\begin{bmatrix} [{}_a\boldsymbol{\omega}\times] & {}_a\boldsymbol{v} \\ \boldsymbol{0} & 0 \end{bmatrix} t\right) \boldsymbol{T}_{a,init}^b\,. \tag{3.24}$$

A more compact notation for (3.24) is obtained by writing the twist in the exponential instead of the corresponding matrix:

$$\boldsymbol{T}_a^b(t) = \exp\left(\mathbf{t}_a^{object}\ t\right) \boldsymbol{T}_{a,init}^b\,. \tag{3.25}$$

Logarithm of a finite displacement

The logarithm of a finite displacement is also a well-defined operation (Murray, Li, and Sastry 1994, p. 414). The result of the logarithm operation on a finite displacement is the screw twist that generates this displacement in one unit of time. When using a homogeneous transformation matrix for the displacement, the logarithm of this matrix gives the screw twist in the form of the argument of the exponential function in 3.24.

3.2 Solving a Set of Linear Equations

Consider a set of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$, with $\boldsymbol{A}$ an $m \times n$ matrix, $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T$ and $\boldsymbol{b} = \begin{bmatrix} b_1 & b_2 & \dots & b_m \end{bmatrix}^T$. This set of m linear equations express linear *constraints* in n variables. Four distinct cases are possible:

- $rank\boldsymbol{A} = rank \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \end{bmatrix} = n.$
  In this case, the linear set is exactly constrained. There is a unique solution $\boldsymbol{x}_s = \boldsymbol{A}^{-1}\boldsymbol{b}$ which complies to all constraints.
- $rank\boldsymbol{A} = rank \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \end{bmatrix} < n.$
  In this case, the set is underconstrained. An infinite number of solutions $\boldsymbol{x}_s$ comply to all constraints. For one of these solutions, the norm $||\boldsymbol{x}_s||$ is minimal.
- $rank\boldsymbol{A} = n$ and $rank\boldsymbol{A} \neq rank \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \end{bmatrix}.$
  This is the overconstrained case: there is no solution which complies fully to all constraints. However, one value for $\boldsymbol{x}$ minimizes the norm of the error $||\boldsymbol{Ax} - \boldsymbol{b}||$.
- $rank\boldsymbol{A} < n$ and $rank\boldsymbol{A} \neq rank \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \end{bmatrix} < n.$
  In this case, the set is both over- and underconstrained. There is no solution which fully complies to all constraints. However, there is an infinite number of vectors $\boldsymbol{x}$ for which $||\boldsymbol{Ax} - \boldsymbol{b}||$ is minimal. Again, only one vector $\boldsymbol{x}$ minimizes both $||\boldsymbol{x}||$ and $||\boldsymbol{Ax} - \boldsymbol{b}||$.

The Moore Penrose pseudo-inverse $\boldsymbol{A}^\dagger$ yields the minimum norm solution $\boldsymbol{x}_s$, minimizing $||\boldsymbol{x}||$ and $||\boldsymbol{Ax} - \boldsymbol{b}||$ (Penrose 1955):

$$\boldsymbol{x}_s = A^\dagger \boldsymbol{b}.$$

3.2.1 Weighted Pseudo-inverses

Such sets of linear equations are of interest in robotics. Often however, the use of the pseudo-inverse to solve these sets of linear equations leads to non-invariant results, as the vector spaces of $\boldsymbol{x}$ and $\boldsymbol{b}$ are not necessarily Euclidean-normed. One particular example is the linear relation between the end effector twist $\mathbf{t}$ and the joint velocities $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$, given by the robot Jacobian $\boldsymbol{J}_{\boldsymbol{R}}$:

$$\mathbf{t} = \boldsymbol{J}_{\boldsymbol{R}} \dot{\boldsymbol{q}}_{\boldsymbol{R}}.$$

Often the desired value for the robot twist is given and the corresponding joint velocities are to be calculated. Depending on the topology and the joint positions of the robot, the four cases above are present in this example:

- Exactly constrained: a 6D twist, performed by a 6 degree of freedom (DOF) robot in a non-singular position (a position for which $\boldsymbol{J_R}$ is not singular, this is, for which $\boldsymbol{J_R}$ has full rank),
- Underconstrained: a 6D twist, performed by a redundant robot (with more than 6 DOF) in a non-singular position,
- Overconstrained: a 6D twist, to be performed by a robot with less than 6 DOF, or by a 6 DOF robot in a singular position,
- Under- and overconstrained: a 6D twist, to be performed by a redundant robot in a singular position.

In the non-exactly constrained cases, it is a common requirement to minimize the joint velocities $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ and the error on the desired twist: $\mathbf{t}_{err} = \mathbf{t} - \boldsymbol{J_R}\dot{\boldsymbol{q}}_{\boldsymbol{R}}$. However, the pseudo-inverse solution minimizes the Euclidean norms $||\dot{\boldsymbol{q}}_{\boldsymbol{R}}||$ and $||\mathbf{t}_{err}||$. As the components of a twist have units of translational and rotational velocity (such as m/s and rad/s), the Euclidean norm of a twist clearly has no physical meaning. Furthermore, the norm is not invariant with respect to the choice of units. For the joint velocities similar remarks are valid, for instance in the case of a robot with rotational and translational joints.

A solution to this problem is to choose weighting matrices $\boldsymbol{W}_x$ and $\boldsymbol{W}_b$ which invariantly define the weighted norms $||\boldsymbol{x}||_{\boldsymbol{W}_x}$ and $||\boldsymbol{b}||_{\boldsymbol{W}_b}$:

$$\begin{aligned} ||\boldsymbol{x}||_{\boldsymbol{W}_x} &= \boldsymbol{x}^T \boldsymbol{W}_x \boldsymbol{x}, \\ ||\boldsymbol{b}||_{\boldsymbol{W}_b} &= \boldsymbol{b}^T \boldsymbol{W}_b \boldsymbol{b}. \end{aligned}$$

In the case of the robotics example above, a possible choice for these weighting matrices is the mass matrix of the robot in jointspace $\boldsymbol{M}_q$ and in Cartesian space $\boldsymbol{M}_t$. With these matrices as weighting matrices, the norms $||\dot{\boldsymbol{q}}_{\boldsymbol{R}}||_{\boldsymbol{M}_q}$ and $||\mathbf{t}_{err}||_{\boldsymbol{M}_t}$ have a physical meaning: they express the kinetic energy of the robot and the kinetic energy in the error on the desired twist.

The *weighted pseudo-inverse* $\boldsymbol{A}^{\#}$ yields the minimum norm solution $\boldsymbol{x}_s$, minimizing $||\boldsymbol{x}_s||_{\boldsymbol{W}_x}$ and $||\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_s||_{\boldsymbol{W}_b}$ (Doty, Melchiorri, and Bonivento 1993):

$$\boldsymbol{A}^{\#} = \boldsymbol{L}_x^{-1}(\boldsymbol{L}_b \boldsymbol{A} \boldsymbol{L}_x^{-1})^{\dagger} \boldsymbol{L}_t,$$

with $\boldsymbol{L}_x$ and $\boldsymbol{L}_b$ matrices complying to:

$$\begin{aligned} \boldsymbol{W}_x &= \boldsymbol{L}_x^T \boldsymbol{L}_x, \\ \boldsymbol{W}_b &= \boldsymbol{L}_b^T \boldsymbol{L}_b. \end{aligned}$$

3.2.2 Nullspace Constraints

In the underconstrained case, extra degrees of freedom remain available to realize subtasks. Next to the control constraints of the task, the *primary* constraints, the programmer can specify a number of extra, *secondary* constraints, which are not *critical* to the task, but which nevertheless are *desirable*. When the sets of primary and secondary constraints are not conflicting, both sets of constraints are completely realized. When they are conflicting however, the set of primary constraints is fully realized, while the secondary constraints are only realized to such an extent that they don't conflict with the primary constraints (Ben-Israel and Greville 1980).

Consider a set of primary constraints $\boldsymbol{Ax} = \boldsymbol{b}$. All solutions realizing the primary constraints are given by:

$$\boldsymbol{x} = \boldsymbol{A}^{\#}\boldsymbol{b} + \left(\boldsymbol{I} - \boldsymbol{A}^{\#}\boldsymbol{A}\right)\boldsymbol{x}_{sec}, \tag{3.26}$$

with $\boldsymbol{x}_{sec}$ an arbitrary vector of the dimension of $\boldsymbol{x}$. As $\left(\boldsymbol{I} - \boldsymbol{A}^{\#}\boldsymbol{A}\right)$ is a projection matrix which projects $\boldsymbol{x}_{sec}$ onto the nullspace of $\boldsymbol{A}$, $\boldsymbol{x}_{sec}$ will not conflict with the primary constraints.

Now consider a set of secondary constraints $\boldsymbol{Cx} = \boldsymbol{d}$. Specific values for $\boldsymbol{x}_{sec}$, realizing these secondary constraints, are obtained by solving $\boldsymbol{x}_{sec}$ from this set of constraints and (3.26):

$$\begin{aligned} \boldsymbol{C}\left(\boldsymbol{A}^{\#}\boldsymbol{b} + \left(\boldsymbol{I} - \boldsymbol{A}^{\#}\boldsymbol{A}\right)\boldsymbol{x}_{sec}\right) &= \boldsymbol{d} \\ \Leftrightarrow \left(\boldsymbol{C} - \boldsymbol{C}\boldsymbol{A}^{\#}\boldsymbol{A}\right)\boldsymbol{x}_{sec} &= \boldsymbol{d} - \boldsymbol{C}\boldsymbol{A}^{\#}\boldsymbol{b} \\ \Rightarrow \boldsymbol{x}_{sec} &= \left(\boldsymbol{C} - \boldsymbol{C}\boldsymbol{A}^{\#}\boldsymbol{A}\right)^{\dagger}\left(\boldsymbol{d} - \boldsymbol{C}\boldsymbol{A}^{\#}\boldsymbol{b}\right). \end{aligned}$$

The corresponding values for $\boldsymbol{x}$ are then given by:

$$\boldsymbol{x} = \boldsymbol{A}^{\#}\boldsymbol{b} + \left(\boldsymbol{I} - \boldsymbol{A}^{\#}\boldsymbol{A}\right)\left(\boldsymbol{C} - \boldsymbol{C}\boldsymbol{A}^{\#}\boldsymbol{A}\right)^{\dagger}\left(\boldsymbol{d} - \boldsymbol{C}\boldsymbol{A}^{\#}\boldsymbol{b}\right). \tag{3.27}$$

In this, the dagger symbol also denotes weighted pseudo-inverse. A different symbol is used here, as both pseudo-inverses in above formulas do not necessarily use the same weighting matrices.

Chapter 4

Task Modeling and Specification

This chapter presents a methodology to specify constraint-based robot tasks. Section 4.1 gives an introduction on constraint-based control. In Section 4.2, an example application is discussed to introduce the main concepts of the methodology, without paying too much attention to the details. A more extensive discussion of the methodology is then given in Section 4.3.

4.1 Introduction

This thesis presents iTASC (instantaneous Task Specification using Constraints), a task specification methodology: 1) for *sensor-based* robot tasks; 2) which can deal with more *complex tasks* than the current state of the art (for instance, than the TFF or the approaches based on the definition of wrench and twist bases); 3) which is *general*, in that it does not focus on one particular kind of task, sensor or robot system; and 4) which can deal with *geometric uncertainty*, this is, which provides explicit support for *estimation* of geometric parameters. iTASC is a *constraint-based* methodology, and uses an *instantaneous optimization technique* in the under- or overconstrained case.

In the context of this thesis, complex tasks are those tasks consisting of multiple concurrent subtasks. Such concurrent subtasks are for instance motion specifications for multiple cooperating robots or for different parts of a single robot, or sensor specifications concerning different sensors which concurrently provide information needed to realize the task. Consider for instance a task where a humanoid robot is standing next to a table, holding an object in its hands. The goal is to place the object at a specific location on the table. Cameras are installed inside the robot's head to identify the desired position for the object, and wrist-mounted force sensors provide information about the contact force when placing down the object. The task consists (mainly) of the following subtasks:

- Point the cameras (or thus the head) at the table so the desired location for the object can be identified,
- Align the object with its desired location, based on the camera measurements,
- Place down the object using force control,
- While doing above specifications, keep the center of gravity above the robot base, to prevent the robot from falling over.

Such a task involves concurrent motion and sensor specifications, for different parts of the robot and for different sensors. These specifications cannot be expressed in a single task frame (this is, according to the TFF) or in terms of a twist and wrench base. They require a true constraint-based approach, in which a task is specified by general constraints on the robot motion, possibly in different frames or bases, or concerning different parts of the robot or different sensors.

Constraint-based control

Samson, Le Borgne, and Espiau (1991) have put a major step forward in the area of constraint-based programming. They introduce the *Task Function Approach*. The basic idea behind the approach is that a robot control problem boils down to regulating a –possibly multidimensional– function, known as the *Task Function*, which characterizes the task. Hence, a task is specified by the Task Function $\boldsymbol{e}(\boldsymbol{q_R}, t)$ and the desired values $\boldsymbol{e}_{des}$ for this function:

$$\boldsymbol{e}(\boldsymbol{q_R}, t) = \boldsymbol{e}_{des}. \tag{4.1}$$

As (4.1) expresses a set of (possibly nonlinear) constraints on $\boldsymbol{q_R}$, the name *constraint-based programming* is used.

The Task Function Approach explicitly limits the state of the task to the joint positions $\boldsymbol{q_R}$ and the time t. This approach is appropriate for many tasks, as robot tasks can often be analyzed as positioning problems and in that case only depend on $\boldsymbol{q_R}$ and t. For purely geometrical tasks dealing with relative positioning of objects, such as pick-and-place tasks, laser engraving or spray painting, this is clearly the case. Often however, more complex tasks are also solely defined by $\boldsymbol{q_R}$ and t. This is for instance the case for all tasks involving motion based on data from geometrical sensors, such as a camera or a laser distance sensor. And also tasks involving dynamical processes such as force/torque interaction can be described in terms of $\boldsymbol{q_R}$ and t, if these processes happen slowly enough so the assumption of quasistatic behavior is valid (and hence the dependence of the Task Function on $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ and $\ddot{\boldsymbol{q}}_{\boldsymbol{R}}$ can be neglected).

For a velocity resolved robot, the control program generates the joint velocities which are applied to the robot to realize the task. The first order derivative of (4.1) expresses the relation, this is the *constraints*, to which these joint velocities should comply:

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q_R}} \dot{\boldsymbol{q}}_{\boldsymbol{R}} + \frac{\partial \boldsymbol{e}}{\partial t} = \dot{\boldsymbol{e}}_{des}. \tag{4.2}$$

As (4.2) is linear in $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$, the techniques described in Section 3.2 can be used to calculate $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$.

The concept of describing a task in terms of a linear set of constraints (4.2) is simple, but it is not necessarily straightforward to actually specify the constraints. The most generic procedure is to infer (4.1) and its derivative (4.2) analytically. However, this is only practical for simple tasks. To specify more complex tasks (this is, more complex constraints), task specification support is needed. This chapter describes a task specification approach for constraint-based tasks. Extensions such as the estimation of geometrical parameters are discussed in Chapter 5.

Central in the approach is a model of the task. The task model provides the necessary context to make the definition of the constraints (more) straightforward. Also in other task specification approaches a task model is present, though not always explicitly emphasized. For instance, in the TFF the model is given by the Task Frame and the velocity and force controlled directions in that frame. Once these are defined, task specification becomes straightforward: the user only has to specify the desired force or velocity in each direction, to which the actual force or velocity are then regulated. These specifications are in fact also constraints, defined in the context of the Task Frame. To support the specification of more complex constraints, not limited to a single Task Frame or base, this chapter introduces a more extensive task model. To this purpose, the concepts of *objects* and *features* are introduced. Their relative motion is then modeled in terms of relative twist bases or *feature Jacobians*. Finally, constraints are specified in terms of this relative motion.

The objects, features and feature Jacobians are rather abstract concepts when no specific application is considered. Furthermore, the different steps in the methodology are to some extent interwoven. For instance, the choice of the feature Jacobians influences the definition of the constraints. So to fully understand all steps, a complete application must be considered. Therefore, as an introduction to the formalism, Section 4.2 first discusses an example application. The example aims to clarify the main concepts and to give an

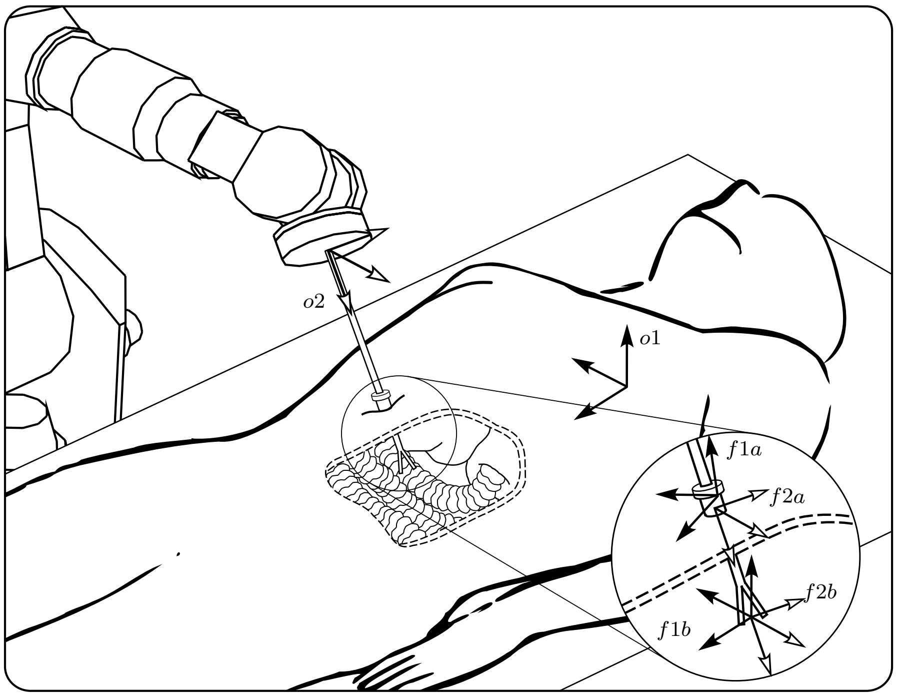


Figure 4.1: The object and feature frames for a minimally invasive surgery task.

overview of all steps of the procedure, without paying too much attention to the details. A more extensive discussion of the methodology follows in Section 4.3.

4.2 An Illustrative Example

This section describes an example task to introduce the concepts of the iTASC methodology. The methodology consists of four steps:

1. Choice of the object and features,
2. Modeling of the relative motion of the objects and features,
3. Definition of the constraints,
4. Solving for the instantaneous motion.

The example application is a minimally invasive surgery task (Figure 4.1). A six DOF robot holds a laparascopic tool, which has a gripper as its end effector. The tool is inserted into a patient's body through a hole, called the trocar. Hence, the robot's motion is to be controlled such that the trocar point, this is, the intersection point of the tool and the patient, stays at its desired position. Furthermore, the robot's motion should be such that the endpoint of the tool moves along a specified trajectory to reach an organ in the patient's body. This suggests two motion constraints: one regarding the position of the trocar point, and one regarding the motion of the endpoint of the tool.

4.2.1 Step 1: Choice of the Objects and Features

The first step of the task specification methodology concerns the choice of the objects and features relevant to the task. As the minimally invasive surgery task deals with motion of the laparascopic tool with respect to the patient, the two relevant objects are the patient's body (object 1), and the tool (object 2). The tool is attached to the robot mounting plate. The features are those parts of both objects which are associated to a motion constraint. In the surgery task, there are two constraints and hence two features: the trocar point (feature a) and the endpoint of the tool (feature b). Note that each of the features is related to both of the objects:

a: The trocar point is the intersection point of the tool and the patient's body. Hence, at every timestep one specific point of the tool coincides with the trocar point, as well as one specific point on the patient's body. Furthermore, in every timestep the desired location of the trocar is at one specific point of the patient's body.

b: The endpoint of the tool is a fixed point on the tool, and should move along a specified trajectory with respect to the patient's body, to reach an organ.

4.2.2 Step 2: Modeling of the Relative Motion

Next, reference frames are introduced which define the pose of the objects and features. Figure 4.1 shows these frames. Two frames $o1$ and $o2$ are attached to object 1 and object 2 respectively, and for each feature x two feature frames $f1x$ and $f2x$ are defined, reflecting that each feature is related to both objects. The frames are chosen as follows:

- Frame $o1$ is fixed at a reference position on the patient's body.
- Frame $o2$ is fixed to the mounting plate of the robot. It has its origin at the attachment point of the laparascopic tool, and it is oriented with its Z-axis along the tool.

- Frame $f1a$ is rigidly attached to $o1$, with its origin at the *desired* position for the trocar point, and its Z-axis normal to the patient's body.
- Frame $f2a$ is located on the laparascopic tool, with its origin at the *actual* trocar point, while its orientation is the same as that of $o2$.
- Both frames $f1b$ and $f2b$ have their origin at the endpoint of the tool. The orientation of $f1b$ is the same as that of $o1$, while the orientation of $f2b$ is the same as that of $o2$.

This is just one particular choice of frames, and it is not the only possible choice. It is however not random either: the frames are chosen such that, for each of the features, the sequence of frames $o1 \rightarrow f1 \rightarrow f2 \rightarrow o2$ forms a kinematic chain. This is, for each feature x the six degrees of freedom between the two object frames are divided over three submotions:

- Submotion I: of $f1x$ with respect to $o1$,
- Submotion $I\!I$: of $f2x$ with respect to $f1x$,
- Submotion $I\!I\!I$: of $o2$ with respect to $f2x$.

For each of these submotions, a base $\boldsymbol{J_{Fi}}$, with $i = I, I\!I$ or $I\!I\!I$, is defined, which spans the corresponding twist space. These bases are called the *feature Jacobians*. The coordinates of the twist in such a base are denoted by column vectors $\boldsymbol{\tau}_i$, called the *feature twist coordinates*:

$$\mathbf{t}_{o1}^{f1} = \boldsymbol{J_{FI}} \boldsymbol{\tau}_I, \quad \mathbf{t}_{f1}^{f2} = \boldsymbol{J_{FI\!I}} \boldsymbol{\tau}_{I\!I}, \quad \mathbf{t}_{f2}^{o2} = \boldsymbol{J_{FI\!I\!I}} \boldsymbol{\tau}_{I\!I\!I}. \tag{4.3}$$

$\boldsymbol{J_{FI}}$, $\boldsymbol{J_{FI\!I}}$ and $\boldsymbol{J_{FI\!I\!I}}$ are $6 \times n_i$ matrices, where n_i is the number of degrees of freedom of submotion i. Any base can be used for $\boldsymbol{J_{Fi}}$, as long as it spans the motion space of the submotion i. However, every specific choice for a base gives the corresponding feature twist coordinates a certain meaning. For instance, when a twist base vector corresponds to a unit twist along the X-axis of a frame, the corresponding feature twist coordinate reflects the x-velocity with respect to that frame. When an appropriate choice is made for the feature Jacobians, the constraints are easily expressed in terms of the feature coordinates (Section 4.2.3). Furthermore, an adequate choice for the reference frame and reference point of the base leads to easy expressions of the base vectors in the feature Jacobians. In the minimally invasive surgery task, the following feature Jacobians are chosen for feature a:

- There is no motion between $f1a$ and $o1$:

$$\begin{aligned} \mathbf{t}_{o1}^{f1a} &= \boldsymbol{J}_{\boldsymbol{FI}}^{a} {\boldsymbol{\tau}_I}^{a}, \\ &= \mathbf{0}. \end{aligned} \tag{4.4}$$

- The curvature of the patient's skin around the trocar point is moderate, and only small deviations in the position of the actual and the desired trocar point are possible. Furthermore, for these small deviations, the deflection normal to the patients skin will be negligible. Because of this, the motion of the trocar point (this is, of $f2a$) can be modeled as taking place in a plane which is tangential to the patient's body, through the desired trocar point. As $f1a$ is oriented with its Z-axis perpendicular to the patient's body, this plane is given by the XY-plane of $f1a$. Hence $f2a$ has translational degrees of freedom along the X and Y-axes of $f1a$. The first two columns of $\boldsymbol{J}^a_{\boldsymbol{F\mathit{II}}}$ express these degrees of freedom. Furthermore, three rotational degrees of freedom are present between $f2a$ and $f1a$. These are expressed by the last three columns of $\boldsymbol{J}^a_{\boldsymbol{F\mathit{II}}}$:

$$\begin{aligned} {}^{f2a}_{f1a}\mathbf{t}^{f2a}_{f1a} &= \boldsymbol{J}^a_{\boldsymbol{F\mathit{II}}}\boldsymbol{\tau}_{\mathit{II}}{}^a, \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau^a_1 \\ \tau^a_2 \\ \tau^a_3 \\ \tau^a_4 \\ \tau^a_5 \end{bmatrix}. \end{aligned} \tag{4.5}$$

  Note that $\boldsymbol{J}^a_{\boldsymbol{F\mathit{II}}}$ effectively reflects that the motion of $f2a$ is modeled as taking place in the XY-plane of $f1a$: the third row of $\boldsymbol{J}^a_{\boldsymbol{F\mathit{II}}}$ consists of zeros.

- The only possible motion of $f2a$ with respect to $o2$ is translation along the tool, this is, along the Z-axis of $o2$:

$$\begin{aligned} {}^{f2a}_{o2}\mathbf{t}_{o2} &= \boldsymbol{J}^a_{\boldsymbol{F\mathit{III}}}\boldsymbol{\tau}_{\mathit{III}}{}^a, \\ &= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \tau^a_6 \end{bmatrix}. \end{aligned} \tag{4.6}$$

Note that in (4.5) – (4.6) one of the object or feature frames is chosen as reference frame, and one of the origins of these frames as reference point. This, combined with the particular choice for these frames as made previously, leads to easy expressions for these bases: all base vectors are unit vectors along one of the axes of the base frame.

The number of degrees of freedom spanned by the submotions of feature a, this is, the sum of the number of linearly independent columns of the matrices

$\boldsymbol{J}^{a}_{\boldsymbol{F}i}$ is equal to six[1]. This is a logical consequence of the sequence of frames $o1 \rightarrow f1 \rightarrow f2 \rightarrow o2$ being a kinematic chain, which models the six degrees of freedom between $o1$ and $o2$. The same is true for the feature Jacobians for feature b, which are chosen as follows:

- $f1b$ has three translational degrees of freedom with respect to $o1$:

$$\begin{aligned} {}^{f1b}_{o1}\mathbf{t}^{f1b}_{o1} &= \boldsymbol{J}^{b}_{\boldsymbol{F}I}\boldsymbol{\tau}_{I}{}^{b}, \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tau^{b}_{1} \\ \tau^{b}_{2} \\ \tau^{b}_{3} \end{bmatrix}. \end{aligned} \tag{4.7}$$

- $f2b$ has three rotational degrees of freedom with respect to $f1b$:

$$\begin{aligned} {}^{f2b}_{f1b}\mathbf{t}^{f2b}_{f1b} &= \boldsymbol{J}^{b}_{\boldsymbol{F}II}\boldsymbol{\tau}_{II}{}^{b}, \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau^{b}_{4} \\ \tau^{b}_{5} \\ \tau^{b}_{6} \end{bmatrix}. \end{aligned} \tag{4.8}$$

- There is no relative motion of $f2b$ with respect to $o2$:

$$\begin{aligned} \mathbf{t}^{f2b}_{o2} &= \boldsymbol{J}^{b}_{\boldsymbol{F}III}\boldsymbol{\tau}_{III}{}^{b}, \\ &= \mathbf{0}. \end{aligned} \tag{4.9}$$

Note that also for feature b the reference frames and reference points were chosen such, that the feature Jacobians are easily expressed mathematically: also for feature b, the base vectors are unit vectors along one of the frame axes. This is a general rule of thumb when choosing the frames. As much as possible, the frames are aligned so that their axes correspond to a motion degree of freedom of the features.

As $o2$ is attached to the robot mounting plate, its twist $\mathbf{t}^{o2}_{w}$ can be expressed in terms of the robot Jacobian $\boldsymbol{J_R}$ and joint velocities $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$:

$$_{w}\mathbf{t}^{o2}_{w} = \boldsymbol{J_R}\dot{\boldsymbol{q}}_{\boldsymbol{R}}. \tag{4.10}$$

[1] As the feature Jacobians are twist bases, their columns are twists. Hence, to check the linear independence, the twists should be transformed to a common reference frame and reference point.

In this, w is the *world* frame, an inertial reference frame with respect to which the robot Jacobian is expressed. The twist of $o1$ depends on the movements of the patient and is not controllable by the robot. In a real setup, sensor measurements of the patient's movements are needed to obtain values for $\mathbf{t}_w^{o1}$. Possibly, also a motion model is available. For instance, the respiratory motion of the upper body is quite repetitive so this motion can be modeled. Estimation techniques such as the Kalman Filter use such models to generate predictions, which are corrected using the measurements. This yields a better estimate of the real motion than the use of the raw measurements. In the context of this example however, it is not relevant how the values for $\mathbf{t}_w^{o1}$ are obtained.

The definition of the feature Jacobians and the description of the motion of $o1$ and $o2$ conclude the second step of the task specification methodology.

4.2.3 Step 3: Definition of Constraints

In the third step, constraints are defined to impose the desired motion. Two sets of constraints are needed: one regarding the position of the trocar point, and one regarding the motion of the endpoint of the tool:

- τ_1^a and τ_2^a express the translational X and Y-velocity of $f2a$ with respect to $f1a$, or thus of the actual trocar point with respect to its desired position. To realize the desired position of the trocar two constraints are expressed, imposing the values of τ_1^a and τ_2^a to the outcome of a position controller:

$$\tau_1^a = k_{fb}(x^a_{desired} - x^a_{actual}), \tag{4.11}$$

$$\tau_2^a = k_{fb}(y^a_{desired} - y^a_{actual}). \tag{4.12}$$

  In this, k_{fb} is a feedback constant, x^a_{actual} and y^a_{actual} are the coordinates of the origin of $f2a$, expressed in $f1a$, and $x^a_{desired}$ and $y^a_{desired}$ the desired values for these coordinates. As no motion of the trocar is desired, $x^a_{desired}$ and $y^a_{desired}$ are constants. It is of course possible to use different feedback constants for the x and y-direction, or a different type of controller. This is however not relevant in the scope of this example.

- Three other position constraints are needed to impose the translational motion of the endpoint of the tool. These are similar to the previous constraints, but now concern τ_1^b, τ_2^b and τ_3^b, as these coordinates express the X, Y and Z-velocity of $f1b$ with respect to $o1$, or thus of the endpoint of the tool with respect to the patient's body:

$$\begin{aligned}\tau_1^b &= k_{fb}(x^b_{desired} - x^b_{actual}), & (4.13)\\ \tau_2^b &= k_{fb}(y^b_{desired} - y^b_{actual}), & (4.14)\\ \tau_3^b &= k_{fb}(z^b_{desired} - z^b_{actual}). & (4.15)\end{aligned}$$

In this, x^b_{actual}, y^b_{actual} and z^b_{actual} correspond to the x, y and z-coordinates of $f1b$, expressed in $o1$. The desired values for these coordinates, $x^b_{desired}$, $y^b_{desired}$ and $z^b_{desired}$, are variable. Typically, these values are generated by a path planner, which calculates a path from the initial position to a desired final position.

Similarly to those of the feature Jacobians, the mathematical expressions for the constraints are influenced by the choice of the object and feature frames. In this example, the particular choice for the frames yields simple expressions for the constraints: each constraint is expressed in terms of a single feature twist coordinate τ. This is another rule of thumb to choose the frames: the frames are chosen such that the constraints are easily expressed mathematically (for instance related to a single coordinate τ).

4.2.4 Step 4: Solving for the Instantaneous Motion

In this last step the set of constraints (4.11)–(4.15) is combined with the equations of the relative motion (4.4)–(4.10) to obtain a set of constraints (4.2). Finally, this set is solved for the joint velocities $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$, which are then applied to the robot.

Define $\boldsymbol{J}^a_{\boldsymbol{F}}$ and $\boldsymbol{\tau}^a$ as:

$$\begin{aligned}\boldsymbol{J}^a_{\boldsymbol{F}} &= \left[\ {}_w\boldsymbol{J}^a_{\boldsymbol{FI}}\ \middle|\ {}_w\boldsymbol{J}^a_{\boldsymbol{FII}}\ \middle|\ {}_w\boldsymbol{J}^a_{\boldsymbol{FIII}}\ \right], & (4.16)\\ \boldsymbol{\tau}^a &= \left[\ \tau_1^a\ \ \tau_2^a\ \ \ldots\ \ \tau_6^a\ \right]^T. & (4.17)\end{aligned}$$

In this, ${}_w\boldsymbol{J}^a_{\boldsymbol{F}i}$ denotes the twist base $\boldsymbol{J}^a_{\boldsymbol{F}i}$, but with w as reference frame and the origin of w as reference point. For instance, $\boldsymbol{J}^a_{\boldsymbol{FII}}$ was defined in (4.5) with $f1a$ as reference frame and the origin of $f2a$ as reference point. Hence:

$${}_w\boldsymbol{J}^a_{\boldsymbol{FII}} = {}^{f1a}_{w}\boldsymbol{PM}^{f2a}_{w}\boldsymbol{J}^a_{\boldsymbol{FII}}. \qquad (4.18)$$

$\boldsymbol{J}^b_{\boldsymbol{F}}$ and $\boldsymbol{\tau}^b$ are defined in a similar way.

For both features, $\mathbf{t}^{o2}_{o1} = \mathbf{t}^{f1}_{o1} + \mathbf{t}^{f2}_{f1} + \mathbf{t}^{o2}_{f2}$. With (4.3), (4.16) and (4.17), this leads to:

$${}_w\mathbf{t}^{o2}_{o1} = \boldsymbol{J}^a_{\boldsymbol{F}}\boldsymbol{\tau}^a, \qquad (4.19)$$

and also to a similar expression for feature b:

$${}_w\mathbf{t}^{o2}_{o1} = \boldsymbol{J}^b_{\boldsymbol{F}}\boldsymbol{\tau}^b. \qquad (4.20)$$

Consider now the closed kinematic loop $w \to o1 \to f1 \to f2 \to o2 \to w$. The *twist closure equation* for this loop is given by:

$$\mathbf{t}_w^{o1} + \mathbf{t}_{o1}^{f1} + \mathbf{t}_{f1}^{f2} + \mathbf{t}_{f2}^{o2} + \mathbf{t}_{o2}^{w} = \mathbf{0},$$

or:

$$\mathbf{t}_w^{o1} + \mathbf{t}_{o1}^{o2} + \mathbf{t}_{o2}^{w} = \mathbf{0}.$$

Combined with (4.10), (4.19) and (4.20), this yields:

$$\begin{aligned} {}_w\mathbf{t}_w^{o1} + \boldsymbol{J}_{\boldsymbol{F}}^a \boldsymbol{\tau}^a - \boldsymbol{J}_{\boldsymbol{R}} \dot{\boldsymbol{q}}_{\boldsymbol{R}} &= \mathbf{0}, \\ {}_w\mathbf{t}_w^{o1} + \boldsymbol{J}_{\boldsymbol{F}}^b \boldsymbol{\tau}^b - \boldsymbol{J}_{\boldsymbol{R}} \dot{\boldsymbol{q}}_{\boldsymbol{R}} &= \mathbf{0}. \end{aligned}$$

In matrix notation, this is written as:

$$\begin{bmatrix} \boldsymbol{J}_{\boldsymbol{F}}^a & \mathbf{0} \\ \mathbf{0} & \boldsymbol{J}_{\boldsymbol{F}}^b \end{bmatrix} \begin{bmatrix} \boldsymbol{\tau}^a \\ \boldsymbol{\tau}^b \end{bmatrix} - \begin{bmatrix} \boldsymbol{J}_{\boldsymbol{R}} \\ \boldsymbol{J}_{\boldsymbol{R}} \end{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{R}} = - \begin{bmatrix} {}_w\mathbf{t}_w^{o1} \\ {}_w\mathbf{t}_w^{o1} \end{bmatrix},$$

or, with obvious definitions of $\bar{\boldsymbol{J}}_{\boldsymbol{F}}$, $\bar{\boldsymbol{\tau}}$, $\bar{\boldsymbol{J}}_{\boldsymbol{R}}$ and $\bar{\boldsymbol{T}}_u$:

$$\bar{\boldsymbol{J}}_{\boldsymbol{F}} \bar{\boldsymbol{\tau}} - \bar{\boldsymbol{J}}_{\boldsymbol{R}} \dot{\boldsymbol{q}}_{\boldsymbol{R}} = \bar{\boldsymbol{T}}_u. \tag{4.21}$$

Since $\bar{\boldsymbol{J}}_{\boldsymbol{F}}$ is always of full rank, as it is composed of $\boldsymbol{J}_{\boldsymbol{F}}^a$ and $\boldsymbol{J}_{\boldsymbol{F}}^b$ which are full rank bases, (4.21) leads to an expression for $\bar{\boldsymbol{\tau}}$:

$$\bar{\boldsymbol{\tau}} = \bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1} \left(\bar{\boldsymbol{T}}_u + \bar{\boldsymbol{J}}_{\boldsymbol{R}} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \right). \tag{4.22}$$

Defining $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{C}}_{\boldsymbol{F}}$ as:

$$\begin{aligned} \bar{\boldsymbol{u}} &= \begin{bmatrix} k_{fb}(x^a_{desired} - x^a_{actual}) \\ k_{fb}(y^a_{desired} - y^a_{actual}) \\ k_{fb}(x^b_{desired} - x^b_{actual}) \\ k_{fb}(y^b_{desired} - y^b_{actual}) \\ k_{fb}(z^b_{desired} - z^b_{actual}) \end{bmatrix}, \\ \bar{\boldsymbol{C}}_{\boldsymbol{F}} &= \left[\begin{array}{cc|cc} \boldsymbol{I}_{2\times 2} & \boldsymbol{0}_{2\times 4} & \boldsymbol{0}_{2\times 3} & \boldsymbol{0}_{2\times 3} \\ \boldsymbol{0}_{3\times 2} & \boldsymbol{0}_{3\times 4} & \boldsymbol{I}_{3\times 3} & \boldsymbol{0}_{3\times 3} \end{array} \right], \end{aligned}$$

the set of constraints (4.11)–(4.15) is rewritten as:

$$\bar{\boldsymbol{C}}_{\boldsymbol{F}} \bar{\boldsymbol{\tau}} = \bar{\boldsymbol{u}}. \tag{4.23}$$

In this case, $\bar{\boldsymbol{C}}_{\boldsymbol{F}}$ is a *selection matrix*, selecting coordinates τ_i for each constraint, from the complete coordinate vector $\bar{\boldsymbol{\tau}}$. Combined with (4.22), this yields the linear set in $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$:

$$\bar{\boldsymbol{C}}_{\boldsymbol{F}} \bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1} \left(\bar{\boldsymbol{T}}_u + \bar{\boldsymbol{J}}_{\boldsymbol{R}} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \right) = \bar{\boldsymbol{u}},$$

or:

$$\left(\bar{\boldsymbol{C}}_{\boldsymbol{F}}\bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\bar{\boldsymbol{J}}_{\boldsymbol{R}}\right)\dot{\boldsymbol{q}}_{\boldsymbol{R}} = \left(\bar{\boldsymbol{u}} - \bar{\boldsymbol{C}}_{\boldsymbol{F}}\bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\bar{\boldsymbol{T}}_{u}\right). \tag{4.24}$$

In each timestep, the robot joint velocities are solved from this set of equations using the pseudo-inverse technique described in Section 3.2, and applied to the robot. As in this example only five constraints are defined, the set is underconstrained and weights in joint space are needed. For a 6DOF industrial robot with revolute joints, as considered in the example, it is common that the first three axes, at the base of the robot, have a much higher inertia than the last three axes. A possible choice for the weighting matrix reflecting this is for instance $\boldsymbol{W}_q = \text{diag}(10, 10, 10, 1, 1, 1)$. These weights express a 10 times higher cost for velocities of the first three axes of the robot, than for those of the last three axes. Hence, motion of the last three axes is preferred over motion of the first three axes. Note that these proposed weights have no units and are hence not generally applicable, for instance for a robot with both rotational and translational joints. Another, more general choice for the weighting matrix is the real mass matrix of the robot. If the mass matrix is chosen as weighting matrix, the obtained values of $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ correspond to those values which realize the task constraints, while minimizing the instantaneous kinetic energy of the robot.

4.2.5 Experiment

An experiment of the minimally invasive surgery task was performed, though in a more extended setting than explained in the example. Figure 4.2 shows two screenshots of the experiment. One robot is holding a box with a hole in it, simulating the patient. The other robot is holding the laparascopic tool. The first robot is moved around by interacting with a 6D joystick, to simulate motion of the patient. This motion is measured by a Krypton K600 measurement system (Metris 2007), and fed back to the second robot's controller in realtime (more formally: ${}_{w}\mathbf{t}_{w}^{o1}$ is measured by the K600 measurement system). The specifications were to keep the laparascopic tool centered at the trocar (this is, the hole) at all times, and to perform a translational motion of the tip of the tool, along a line inside the box.

4.2.6 Conclusions

An example application was discussed in this section as an introduction to the task specification formalism. While the example provides only a limited overview of the methodology (for instance, no sensor integration is discussed), all different steps of the methodology are present, and the example shows the relations between the different steps. For instance, by choosing appropriate object and feature frames, it becomes easy to specify the feature Jacobians.

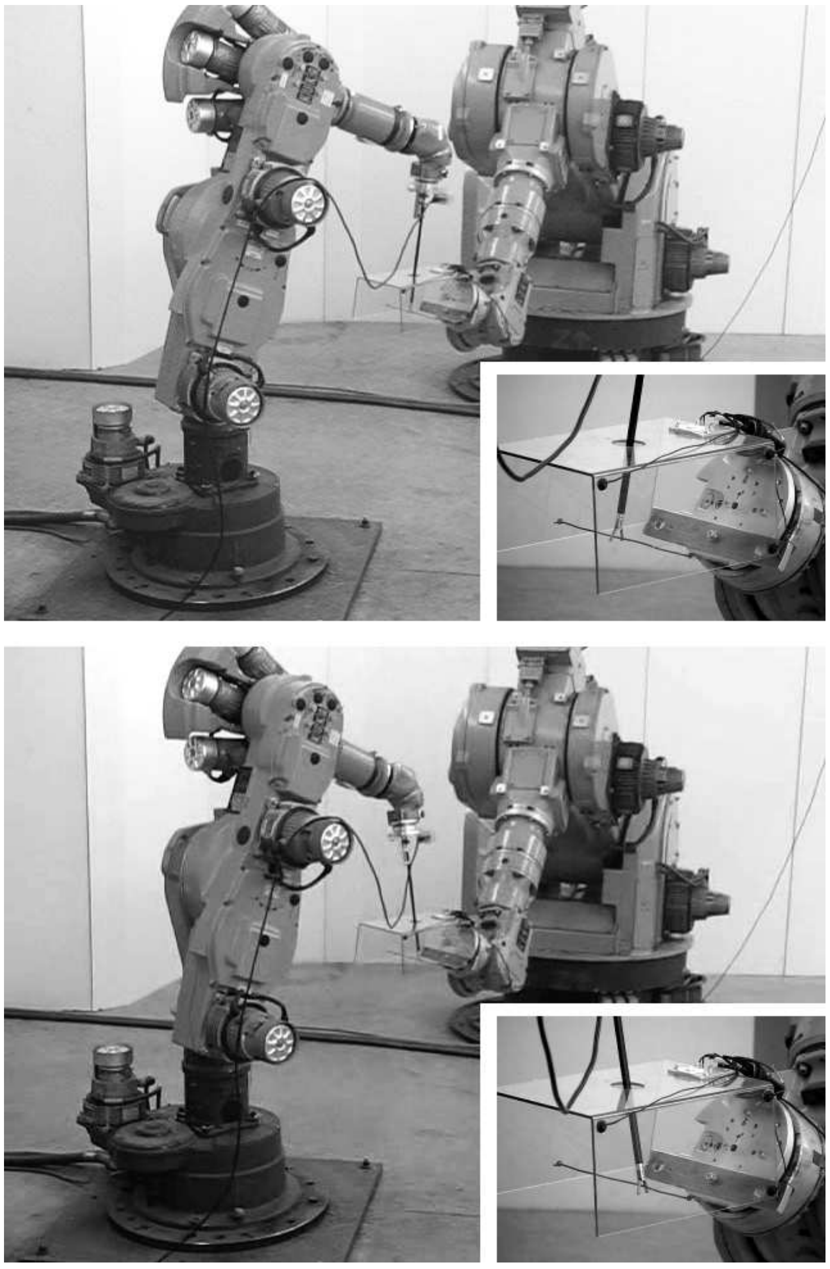


Figure 4.2: Two consecutive snapshots of the minimally invasive surgery experiment. One robot moves a box with a hole in it, simulating motions of the patient. The other robot is holding the laparascopic tool. The specifications are to keep the laparascopic tool centered at the hole at all times, and to perform a translational motion of the tip of the tool along a line inside the box. The insets show a closeup of the tip of the tool.

Furthermore, the chosen example shows the application of the methodology to specify a task which is not easily specified using other specification approaches. For instance, as the example application has two sources of constraints (the motion of the trocar point and the motion of the endpoint of the tool), both the TFF and the approaches based on the definition of an instantaneous twist base are shortcoming.

4.3 Task Modeling and Specification

After the example application, this section now follows with a more in-depth description of the iTASC methodology.

As introduced in Section 4.1, iTASC is a constraint-based approach to task specification, based on the Task Function Approach (Samson, Le Borgne, and Espiau 1991). Recall that according to the Task Function Approach, a robot task is described as a general vector function $\boldsymbol{e}(\boldsymbol{q_R}, t)$ and the desired values $\boldsymbol{e}_{des}$ for this function:

$$\boldsymbol{e}(\boldsymbol{q_R}, t) = \boldsymbol{e}_{des}. \tag{4.25}$$

For velocity resolved robots, the first order derivative of the task function is needed, as this derivative expresses the constraints to which the joint velocities have to comply in order to realize the Task Function:

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q_R}} \dot{\boldsymbol{q}}_{\boldsymbol{R}} + \frac{\partial \boldsymbol{e}}{\partial t} = \dot{\boldsymbol{e}}_{des}, \tag{4.26}$$

As it is for most tasks not straightforward to infer (4.25) and its derivative (4.26) analytically, this chapter introduces the iTASC methodology to support the specification of a set of constraints (4.26).

The methodology consists of building up a task model, and then specifying the constraints in terms of this model. The goal of the task model is to support (or *simplify*) the specification of the constraints. This is similar to for instance the TFF: once a good task frame is chosen for a certain task, the task is easily specified in terms of desired forces or velocities along the axes of that frame. However, in an ill-chosen task frame the specification is less straightforward.

As previously stated, the Task Function Approach primarily deals with geometrical tasks, although dynamical tasks such as force/torque interaction are also possible, if these happen quasistatically. For each set of specifications, this is, each set of constraints, the relevant objects are considered. Then, features are introduced, which closer reflect those parts of the objects which are involved by the constraints. Consequently, the 6 DOFs between the objects are divided over three submotions: two of the features with respect to the objects, and one of the features with respect to each other. The task model describes these motions in terms of feature Jacobians and feature twist coordinates. Finally, the constraints are specified in terms of these feature twist coordinates.

The methodology consists of the following four steps, which are clarified in the rest of this section:

1. Choice of the objects and features,
2. Modeling of the relative motion of the objects and features,
3. Definition of the constraints,
4. Solving for the instantaneous motion.

4.3.1 Step 1: Choice of the Objects and Features

Task relations and relevant objects

In general, the Task Function (4.25), or its first-order expression (4.26), describes a number of concurrent subtasks, in each of which *a certain relation* between *objects* in the robot environment is to be controlled *by moving the robot*. Such a relation is called a *task relation*. Consider for instance the previously mentioned example of a humanoid robot with head-mounted cameras, which should place a box on a table using force control and visual servoing (Section 4.1). Besides others (such as keeping balanced) two subtasks are present in the application:

1. The relative orientation of the head of the robot and the table should be regulated such that the table is centered in the camera images,
2. The robot should move the box such that it is placed at its desired location on the table.

In the first subtask, the two relevant objects are the robot's head and the table, and the task relation is given by their relative orientation. In the second subtask, the relevant objects are the box and the table, and the task relation is their relative pose and the contact force.

Figures 4.3 and 4.4 show two other examples. The first example task (Figure 4.3) involves a forming process: a plate is placed between two spheres, by a peripheral mechanism in a robot workcell. The robot then rolls one of the spheres over the other while maintaining a contact force. This deforms the plate into a curved surface. The objects in this task are the two spheres, one of which is moved by the robot[2]. The task relations concerning these objects

[2]The plate is not a relevant object here, as, concerning the specification of the robot motion, it does not matter whether there is a plate between the spheres or not.

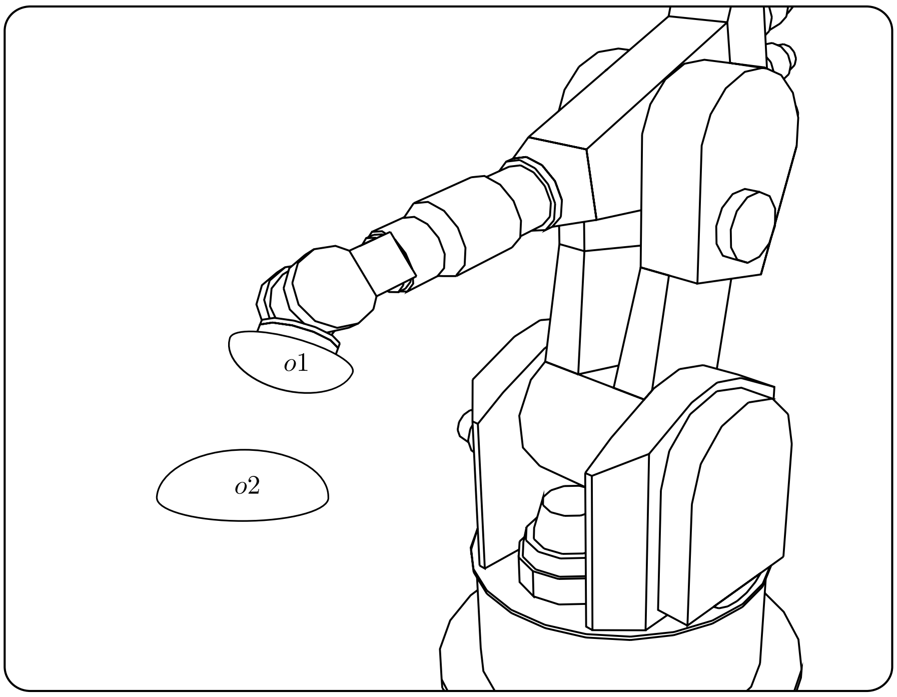


Figure 4.3: A robot performs a forming task, in which a plate (not shown) is placed between two spheres. The robot then rolls one of the spheres over the other while maintaining a contact force. This deforms the plate into a curved surface. The objects in this task are the two spheres, one of which is moved by the robot. The task relations concerning these objects are the contact force between them, and the motion on their surfaces of the contact point.

are the contact force between them, and the motion on their surfaces of the contact point. In the second example task (Figure 4.4), a maintenance robot enters a pipeline through an uncovered flange and points a camera at a seam to perform a visual check. In this task, there are two relevant pairs of objects: one for the relative positioning of the camera and the seam, and another one related to the link which sits through the flange. Possibly however, other objects can be defined, for instance for collision avoidance purpose. In all cases, the pipe is clearly one of the relevant objects, as the task deal with relative positioning of the robot and the pipe. Concerning the positioning of camera relative to the seam, the last link of the robot is the other object, as the camera is attached to this link. The task relation is the relative pose of the camera and the seam. Considering the link which sits through the flange,

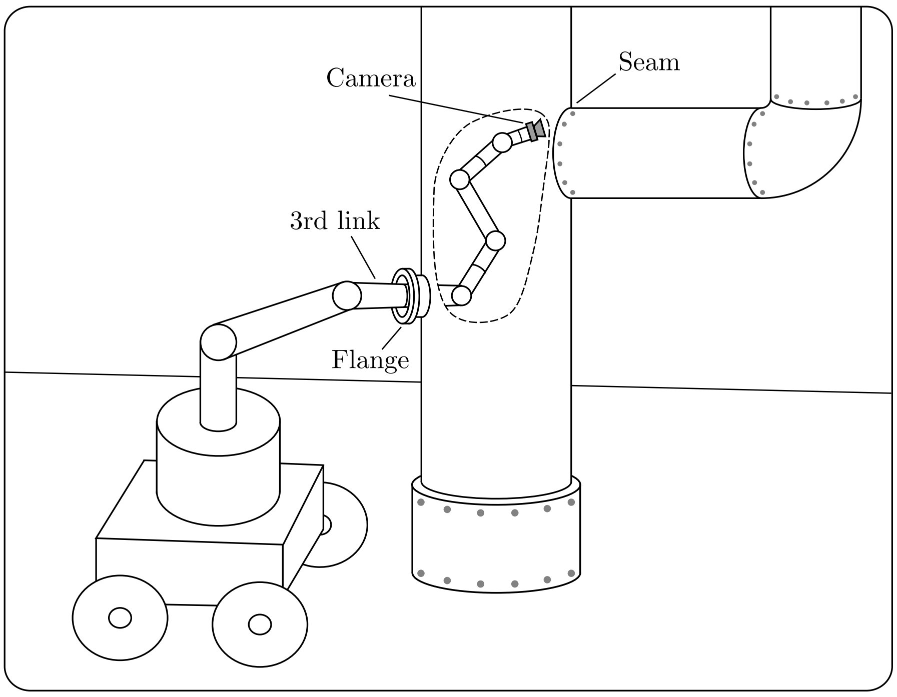


FIGURE 4.4: A maintenance robot enters a pipe through a flange to perform a visual check of a seam. In this task, two pairs of objects are relevant. In both cases, the pipe is one of the relevant objects. The other objects are the last link of the robot, to which the camera is attached, and the third link of the robot, which sits through the flange. The task relations are the relative orientation of the last link (this is, the camera) and the pipe, and the relative pose of the third link of the robot and the flange.

the link itself is the second object. In this case, the task relation is the pose of the link with respect to the flange[3].

So, to specify a task, first the task relations are identified, as well as the corresponding objects. Note that, for each task relation, at least one of both objects is moved by the robot, as otherwise the task relation is uncontrollable. However, it is possible that the object's motion is only partially controlled by the robot, for instance in the case of an object connected to the robot by means of a passive joint.

[3]Note that, for the depicted configuration, the last link of the robot had to pass through the flange first, then the one but last link, and so on. Each of these phases has the same task relation, but a different link as relevant object.

Features of objects

Now, *features* are chosen. A feature is *that part of the object which is relevant to the task relation.* It can be a physical entity, this is, a physical part of an object such as a vertex, edge, or surface, or it can be an abstract geometric property of a physical entity such as the symmetry axis of a hollow cylinder or the reference frame of a sensor connected to the object.

Figure 4.5 shows two example cases. The first example case is a compliant motion task between two polyhedral objects. The goal (this is, the task relation) is to maintain a vertex-face contact between the objects, and to move the objects such that the contact point follows a desired trajectory in the contacting face. In this case the relevant features are the contacting vertex of the first object and the contact point on the second object. The second example case is a peg-in-hole problem. In this case, the relevant features are for instance the symmetry axes of the peg and hole (which are aligned when the peg is in the hole), or their respective surfaces (which make contact in the assembled state). Note that: 1) a task relation inducts a feature on each of both objects, and 2) the peg-in-hole example shows that there is not necessarily a unique choice of features.

The task relation can be expressed in terms of submotions, of the features with respect to the objects, and of the features with respect to each other. For instance, in the polyhedral compliant motion task above, the 6 DOF between both objects are divided over:

- the motion of the contact point in the contacting face (2 DOF),
- the relative rotation of the objects around the contact point (3 DOF), and
- motion along the contacting normal to make/break the contact (1 DOF).

The constraints to realize the task relation (maintaining the contact and to moving the objects such that the contact point follows a desired trajectory) are directly related to these submotions:

- to regulate the contact force, the motion along the contacting normal should be controlled (1 constraint), and
- the motion of the contact point in the contact face should also be controlled (2 constraints).

This illustrates the actual reason why the features are introduced: features divide the relative motion of the objects in submotions, and, for well-chosen features, the task relation is directly expressible in terms of constraints on these submotions. The remainder of the methodology deals with the modeling of the submotions, and with the expression of the constraints on these motions.

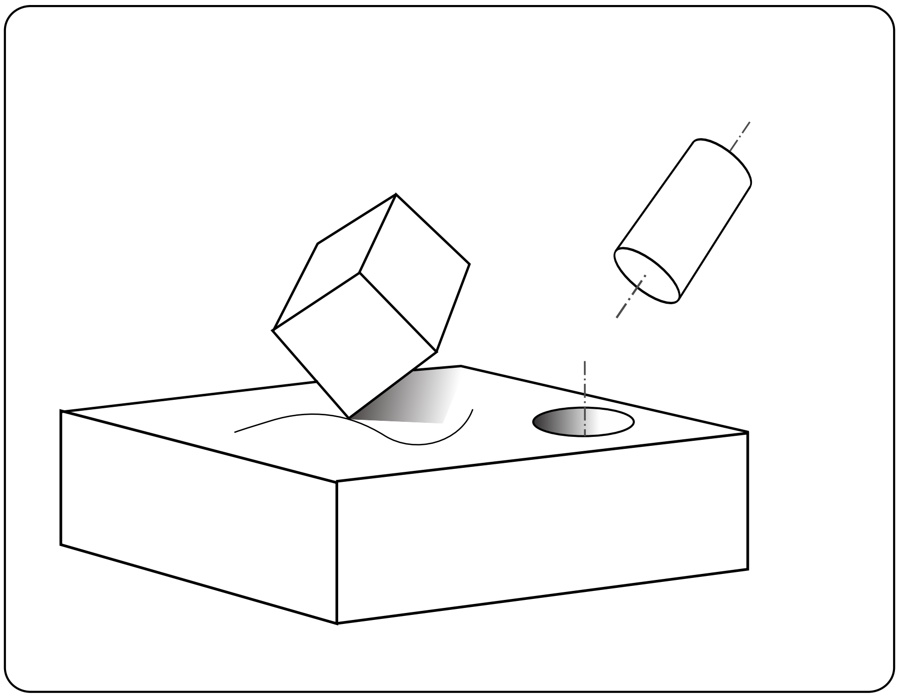


FIGURE 4.5: A feature is linked to an object. It can be a physical entity such as a vertex, edge, face or surface (or thus a physical part of an object), or it can be an abstract geometric property of a physical entity such as the symmetry axis of a hollow cylinder, or the reference frame of a sensor connected to the object (for instance a camera frame).

4.3.2 Step 2: Modeling of the Relative Motion

In the second step of the task specification methodology, frames are introduced as a representation for the objects and features, and the relative motions of the objects and features are modeled in terms of twist bases and coordinates in these bases.

Definition of frames

Each task relation accounts for two objects and two features, so for each task relation four frames are introduced: two object frames $o1$ and $o2$, each attached to one of the objects, and two feature frames $f1$ and $f2$, each attached to one of the corresponding features of the objects. The rules for assigning these frames are:

1. $o1$ and $o2$ are rigidly attached to the corresponding objects,
2. $f1$ and $f2$ are linked, but not necessarily rigidly attached to $o1$ and $o2$ respectively,
3. the connection $o1 \to f1 \to f2 \to o2$ forms a kinematic chain; the six degrees of freedom between $o1$ and $o2$ are distributed over three submotions: submotion I of $f1$ with respect to $o1$, submotion $I\!I$ of $f2$ with respect to $f1$, and submotion $I\!I\!I$ of $o2$ with respect to $f2$.

Furthermore, the frames are chosen such, that the mathematical representation of the submotions is simplified.

For every feature, each of the submotions between $o1$ and $o2$ is represented by an instantaneous twist vector: $\mathbf{t}_{o1}^{f1}$, $\mathbf{t}_{f1}^{f2}$ and $\mathbf{t}_{f2}^{o2}$ respectively. As the six degrees of freedom between $o1$ and $o2$ are distributed over the submotions, each of these twists belongs to a vector space of dimension n_i ($i = I, I\!I$ or $I\!I\!I$), with n_i less than or equal to six and $n_I + n_{I\!I} + n_{I\!I\!I} = 6$. To incorporate this into the model of the relative motion, each twist is parametrized as a set of *feature twist coordinates* $\boldsymbol{\tau}_i$ in a corresponding twist base $\boldsymbol{J_{Fi}}$, called the *feature Jacobian*:

$$\mathbf{t}_{o1}^{f1} = \boldsymbol{J_{FI}}\boldsymbol{\tau}_I, \quad \mathbf{t}_{f1}^{f2} = \boldsymbol{J_{FI\!I}}\boldsymbol{\tau}_{I\!I}, \quad \mathbf{t}_{f2}^{o2} = \boldsymbol{J_{FI\!I\!I}}\boldsymbol{\tau}_{I\!I\!I}. \tag{4.27}$$

The feature Jacobians $\boldsymbol{J_{Fi}}$ are $6 \times n_i$ matrices. As with the choice of the object and feature frames, there are no strict rules to choose a particular base. Moreover, in principle any base which spans the corresponding submotion can be used for $\boldsymbol{J_{Fi}}$. However, the columns of each feature Jacobian express the submotion twist in the case of a unit value for the corresponding coordinate τ and a zero value for the other coordinates. In other words, the chosen feature Jacobian assigns a specific meaning to the coordinates τ. For well chosen feature Jacobians, the coordinates are closely related to the constraints, so the constraints are easily definable in terms of those coordinates. For instance, consider the following feature Jacobian:

$$\begin{aligned} {}_{o1}^{f1}\mathbf{t}_{o1}^{f1} &= \boldsymbol{J_{FI}}\boldsymbol{\tau}_I \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}. \end{aligned} \tag{4.28}$$

This feature Jacobian relates the coordinate τ_1 to the translational velocity of $f1$ (as the origin of $f1$ is the reference point), along the X-axis of $o1$ (as $o1$ is the reference frame). Similarly, τ_2 expresses the translational velocity of $f1$, along the Y-axis of $o1$. Hence, imposing specific values for the coordinates τ_1 yields a certain motion along the X-axis, while constraints on τ_2 yield motion along the Y-axis. In other words, by imposing constraints on the coordinates τ, motion specifications are defined.

The notation ${}_w\boldsymbol{J_{Fi}}$ is used if a feature Jacobian $\boldsymbol{J_{Fi}}$ is expressed with reference point and reference frame w. Define $\boldsymbol{J_F}$ and $\boldsymbol{\tau}$ as:

$$\begin{aligned} \boldsymbol{J_F} &= \left[\ {}_w\boldsymbol{J_{FI}} \mid {}_w\boldsymbol{J_{FII}} \mid {}_w\boldsymbol{J_{FIII}}\ \right], \\ \boldsymbol{\tau} &= \left[\ \boldsymbol{\tau_I} \quad \boldsymbol{\tau_{II}} \quad \boldsymbol{\tau_{III}}\ \right]^T. \end{aligned}$$

Since $\mathbf{t}_{o1}^{o2} = \mathbf{t}_{o1}^{f1} + \mathbf{t}_{f1}^{f2} + \mathbf{t}_{f2}^{o2}$, the following statement holds:

$$_w\mathbf{t}_{o1}^{o2} = \boldsymbol{J_F}\boldsymbol{\tau}. \tag{4.29}$$

At least one of the objects is manipulated by the robot. Furthermore, uncontrolled motion of the objects is possible, for instance if an object is placed on a conveyor belt which is not part of the robot system, or in the case an object is pushed by the robot and the object slips sideways with respect to the pushing direction. Hence, a general description of the object motion is given by:

$$\begin{aligned} \mathbf{t}_w^{o1} &= \boldsymbol{J_{R1}}\dot{\boldsymbol{q}}_{\boldsymbol{R}} + \mathbf{t}_{u_1}, \\ \mathbf{t}_w^{o2} &= \boldsymbol{J_{R2}}\dot{\boldsymbol{q}}_{\boldsymbol{R}} + \mathbf{t}_{u_2}, \end{aligned}$$

in which $\boldsymbol{J_{R1}}$ and $\boldsymbol{J_{R2}}$ express the robot Jacobians for the motion of $o1$ and $o2$ respectively, and $\mathbf{t}_{u_1}$ and $\mathbf{t}_{u_2}$ their uncontrolled twist. Since $\mathbf{t}_{o2}^{o1} = \mathbf{t}_w^{o1} - \mathbf{t}_w^{o2}$, and with introduction of $\boldsymbol{J_R}$ and $\mathbf{t}_u$, the notation is reduced to:

$$\begin{aligned} \mathbf{t}_{o2}^{o1} &= (\boldsymbol{J_{R1}} - \boldsymbol{J_{R2}})\,\dot{\boldsymbol{q}}_{\boldsymbol{R}} + \left(\mathbf{t}_{u_1} - \mathbf{t}_{u_2}\right) \\ &\equiv \boldsymbol{J_R}\dot{\boldsymbol{q}}_{\boldsymbol{R}} + \mathbf{t}_u. \end{aligned} \tag{4.30}$$

Combined with (4.29), for each feature the following velocity closure equation holds:

$$\boldsymbol{J_R}\dot{\boldsymbol{q}}_{\boldsymbol{R}} + \boldsymbol{J_F}\boldsymbol{\tau} + \mathbf{t}_u = \mathbf{0}. \tag{4.31}$$

4.3.3 Step 3: Definition of Constraints

The constraints are now formulated in terms of the robot joint velocities $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ and the feature twist coordinates $\boldsymbol{\tau}$. As the first order expression of the Task Function is considered, as needed for velocity resolved robots, these constraints are linear. In general, the set of constraints is expressed as:

$$\left[\ \boldsymbol{C_R} \mid \boldsymbol{C_F}\ \right] \begin{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \\ \boldsymbol{\tau} \end{bmatrix} = \boldsymbol{u}, \tag{4.32}$$

with $\boldsymbol{C_R}$ and $\boldsymbol{C_F}$ the linear coefficient matrices of respectively $\boldsymbol{\dot{q}_R}$ and $\boldsymbol{\tau}$. Generally, $\boldsymbol{C_R}$ and $\boldsymbol{C_F}$ can be any matrix. The following paragraphs discuss the values of $\boldsymbol{C_R}$ and $\boldsymbol{C_F}$ for three common kinds of constraints:

- Constraints on robot joint velocities or feature frame twists,
- Constraints on robot joint positions or feature frame poses, and
- Constraints on sensor outputs.

Robot joint velocity and feature frame twist constraints

For some applications, it is necessary to directly specify some of the robot joint velocities. For instance, consider a redundant robot with a blocked joint, caused by a hardware failure. The robot can still be used if the velocity of the corresponding joint is set to zero, independently of the other joint velocities. Such cases require explicit constraints on $\boldsymbol{\dot{q}_R}$, as for instance:

$$\begin{aligned}\dot{q}_1 &= u_1,\\ \dot{q}_2 &= u_2,\\ \dot{q}_5 &= u_3.\end{aligned}$$

These three constraints impose desired values u_1, u_2 and u_3 to the joint velocities of axis 1, axis 2 and axis 5 respectively. u_1, u_2 and u_3 are given constants or time functions, specified by the programmer or coming from a path planner. For example, for a 7 DOF robot this yields in the matrix notation of (4.32):

$$\left[\begin{array}{ccccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & \boldsymbol{0}_{3\times 6} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \end{array}\right] \begin{bmatrix} \boldsymbol{\dot{q}_R} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}. \tag{4.33}$$

The three constraints on $\boldsymbol{\dot{q}_R}$ for the 7 DOF robot lead to a 3×7 matrix $\boldsymbol{C_R}$. As (4.33) shows, $\boldsymbol{C_R}$ is a selection matrix for joint velocity constraints: each row contains only zeros, except for one element which is '1' and hence *selects* the corresponding joint velocity. $\boldsymbol{C_F}$ is a zero matrix as there are no constraints on $\boldsymbol{\tau}$.

In a similar way, constraints on feature frame twists are defined by specifying desired values for feature twist coordinates τ. In this case, $\boldsymbol{C_R}$ is the zero matrix and $\boldsymbol{C_F}$ the selection matrix. For instance, for an n DOF robot,

$$\left[\begin{array}{c|cccccc} \boldsymbol{0}_{2\times n} & 1 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right] \begin{bmatrix} \boldsymbol{\dot{q}_R} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} u_4 \\ u_5 \end{bmatrix}$$

expresses desired values u_4 and u_5 for the feature twist coordinates τ_1 and τ_2. For the example feature Jacobian (4.28), these constraints specify desired values u_4 and u_5 for the X and Y-velocity respectively, of $f1$ with respect to $o1$.

Robot joint position and feature frame pose constraints

On a velocity resolved system, positions are realized by a position controller. For instance, for a 1DOF system and a proportional control law, the following constraint realizes a desired position x_{des}:

$$\dot{x} = k_{fb}(x_{des} - x_{meas}).$$

In this, k_{fb} is the feedback constant, x_{meas} the measured position and $\dot{x}$ the commanded velocity.

Similarly, robot joint positions and feature frame poses are realized by imposing the desired values $\boldsymbol{u}$ to correspond to the outcome of a position controller. For joint positions and for feature coordinates which correspond to translational motion, $\boldsymbol{C_R}$ and $\boldsymbol{C_F}$ are again selection matrices. For instance, for the example feature Jacobian (4.28), the following constraints:

$$\left[\begin{array}{cccccc|cccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \begin{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix},$$

with

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} k_{fb1}(q_{1des} - q_{1meas}) \\ k_{fb2}(q_{2des} - q_{2meas}) \\ k_{fb3}(x_{des} - x_{meas}) \end{bmatrix},$$

realize desired positions q_{1des} and q_{2des} for the first two joints, and a desired X-coordinate x_{des} of $f1$ with respect to $o1$[4]

To specify a relative orientation, slightly different constraints are needed. Consider for instance the following feature Jacobian:

$$\begin{aligned} {}^{f1}_{o1}\mathbf{t}^{f1}_{o1} &= \boldsymbol{J_{FI}}\boldsymbol{\tau}_I \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}. \end{aligned} \tag{4.34}$$

According to this feature Jacobian, the feature twist coordinates τ_1, τ_2 and τ_3 correspond to rotational velocities. Such rotational velocities do not integrate

[4] Of course, the choice of controller dictates to what extent the desired positions are actually realized.

into a set of orientation coordinates without integrating factors. Hence, the constraints to specify relative orientations contain integrating factors $\boldsymbol{E}$ (Section 3.1.2, page 26). For instance, for the example feature Jacobian (4.34), the following constraints:

$$\left[\begin{array}{c|c} \boldsymbol{0}_{1\times n} & \boldsymbol{E}^{-1} \end{array}\right] \left[\begin{array}{c} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \\ \boldsymbol{\tau} \end{array}\right] = \left[\begin{array}{c} u_1 \\ u_2 \\ u_3 \end{array}\right],$$

with

$$\left[\begin{array}{c} u_1 \\ u_2 \\ u_3 \end{array}\right] = \left[\begin{array}{c} k_{fb1}(\phi_{des} - \phi_{meas}) \\ k_{fb2}(\theta_{des} - \theta_{meas}) \\ k_{fb3}(\psi_{des} - \psi_{meas}) \end{array}\right],$$

realize a desired orientation $(\phi_{des}, \theta_{des}, \psi_{des})$ of $f1$ with respect to $o1$.

Sensor output constraints

For geometrical sensors, such as cameras or laser distance sensors, the measurement z is function of the relative pose of the objects related to the measurement:

$$\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{d}_{o1}^{o2}).$$

The same kind of measurement equation is valid for dynamical sensors, such as a force/torque sensor, if the assumption of quasistatic behavior is valid. For a velocity resolved system, the first order expression of this equation is considered, which relates the rate of the measurement, $\dot{\boldsymbol{z}}$, to the motion of the objects:

$$\begin{aligned} \dot{\boldsymbol{z}} &= \frac{d\boldsymbol{g}}{d\boldsymbol{d}_{o1}^{o2}} \dot{\boldsymbol{d}}_{o1}^{o2} \\ &= \frac{d\boldsymbol{g}}{d\boldsymbol{d}_{o1}^{o2}} \boldsymbol{\mathcal{E}} \boldsymbol{J}_{\boldsymbol{F}} \boldsymbol{\tau} \\ &\equiv \boldsymbol{J}_s \boldsymbol{\tau}. \end{aligned} \tag{4.35}$$

Equation (4.35) defines the *sensor Jacobian* $\boldsymbol{J}_s$. The control constraint to control a sensor measurement $\boldsymbol{z}$ to its desired value $\boldsymbol{z}_{des}$ is then given by:

$$\left[\begin{array}{c|c} \boldsymbol{0}_{1\times n} & \boldsymbol{J}_s \end{array}\right] \left[\begin{array}{c} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \\ \boldsymbol{\tau} \end{array}\right] = \left[\begin{array}{c} \boldsymbol{u} \end{array}\right],$$

with $\boldsymbol{u}$ the outcome of a control law, such as, in the case of a proportional controller, $\boldsymbol{u} = k_{fb}\,(\boldsymbol{z}_{des} - \boldsymbol{z})$.

The most general way to obtain $\boldsymbol{J}_s$ is to analytically derive $\frac{d\boldsymbol{g}}{d\boldsymbol{\tau}}$. For several practical applications however, $\boldsymbol{J}_s$ is found by inspection. Below, the sensor

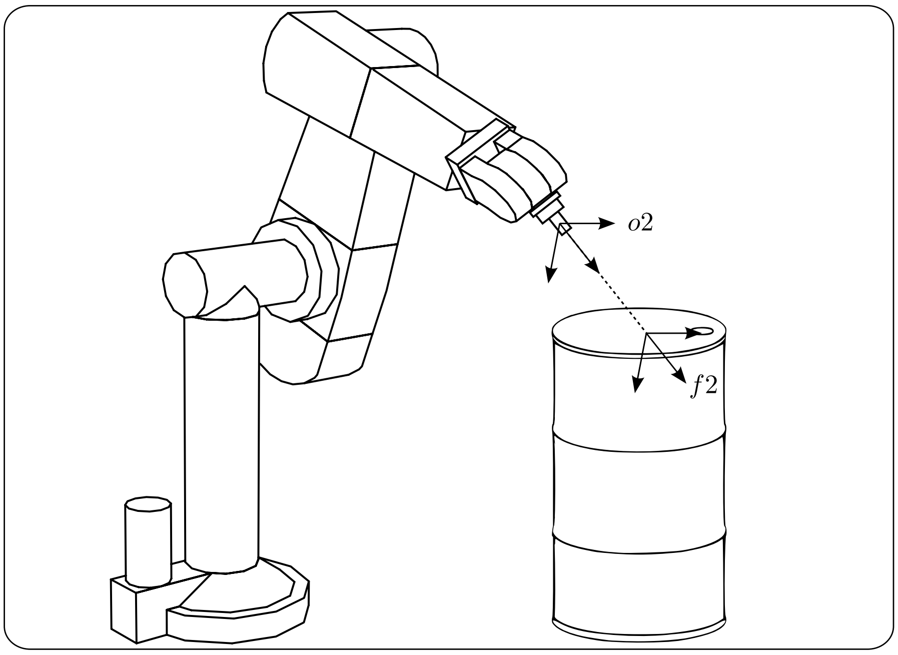


FIGURE 4.6: A robot, pointing a laser distance sensor at a barrel.

Jacobian is derived for three particular examples: a laser distance sensor, a force/torque sensor and a camera, used for 2D visual servoing.

- **Laser distance sensor:** A laser distance sensor measures the distance to an object along a straight line. If one of the feature or object frame axes is aligned with this line, an easy mathematical representation is obtained for the sensor Jacobian. For instance, in the setup shown in Figure 4.6 the $o2$ frame is aligned with its Z-axis along the laser beam. $f2$ is parallel to $o2$, but has its origin at the laser point. The model of motion between those frames is given by:

$$\begin{aligned} {}_{o2}\mathbf{t}_{f2}^{o2} &= \boldsymbol{J}_{\boldsymbol{F}\mathit{III}}\boldsymbol{\tau}_{\mathit{III}} \\ &= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \tau_6 \end{bmatrix}. \end{aligned}$$

  As the sensor measurements correspond to the $Z-$coordinate of the

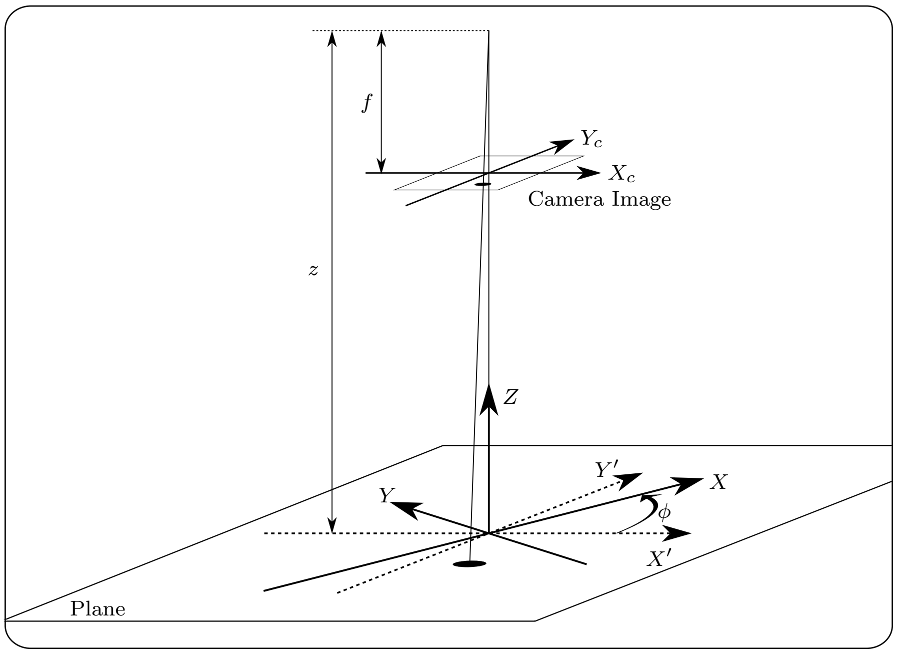


Figure 4.7: Camera projection according to the pinhole camera model.

origin of $f2$ with respect to $o2$, the sensor Jacobian is given by selecting the corresponding row of $\boldsymbol{J_{F_{III}}}$:

$$\boldsymbol{J}_s = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{J_{F_{III}}}.$$

- **Force/torque sensor:** A force/torque sensor measures the forces and torques between contacting objects. A common model for quasistatic force interactions is that of a pure contact stiffness:

$$\boldsymbol{z} = \boldsymbol{K}_{st}\mathbf{t}_{\Delta}.$$

In this, $\boldsymbol{z}$ is the measured wrench, $\mathbf{t}_{\Delta}$ the finite displacement between the objects and $\boldsymbol{K}_{st}$ the contact stiffness. For small displacements $\mathbf{t}_{\Delta}$, $\boldsymbol{K}_{st}$ is constant, which yields:

$$\begin{aligned} \dot{\boldsymbol{z}} &= \boldsymbol{K}_{st}\mathbf{t}_{o1}^{o2}, \\ &= \boldsymbol{K}_{st}\boldsymbol{J_F}\boldsymbol{\tau}. \end{aligned}$$

Hence, in this case the sensor Jacobian is given by $\boldsymbol{K}_{st}\boldsymbol{J_F}$.

- **Camera:** Consider a visual servoing application, in which a camera is moved parallel to a plane. Hence, the camera has 4 DOF with respect to the plane (see figure 4.7): translation in three directions, with coordinates x, y and z, and rotation along the plane normal, this is, along the Z-axis, with coordinate ϕ. The measurements are the coordinates in the camera image of a point in the plane:

  $$\boldsymbol{z} = \begin{bmatrix} x_c \\ y_c \end{bmatrix}.$$

  The focal distance f relates these to the coordinates in the plane:

  $$\begin{aligned} x_c &= \frac{x'}{z} f, \\ y_c &= \frac{y'}{z} f, \end{aligned}$$

  with:

  $$\begin{aligned} x' &= x \cos\phi - y \sin\phi, \\ y' &= x \sin\phi + y \cos\phi. \end{aligned}$$

  Hence:

  $$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \end{bmatrix} = \boldsymbol{J_C} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ 0 \\ 0 \\ \omega_z \end{bmatrix},$$

  with $\boldsymbol{J_C}$ the camera Jacobian:

  $$\boldsymbol{J_C} = \frac{f}{z} \begin{bmatrix} \cos\phi & -\sin\phi & \dfrac{x\cos\phi - y\sin\phi}{z} & 0 & 0 & -\dfrac{x\sin\phi + y\cos\phi}{z} \\ \sin\phi & \cos\phi & -\dfrac{x\sin\phi + y\cos\phi}{z} & 0 & 0 & \dfrac{x\cos\phi - y\sin\phi}{z} \end{bmatrix}.$$

  Since:

  $$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ 0 \\ 0 \\ \omega_z \end{bmatrix} = \boldsymbol{J_F}\boldsymbol{\tau},$$

  this yields the sensor Jacobian:

  $$\boldsymbol{J}_s = \boldsymbol{J_C}\boldsymbol{J_F}.$$

The expression for the complete 6 DOF camera Jacobian can be found in (Espiau, Chaumette, and Rives 1992).

4.3.4 Step 4: Solving for the Instantaneous Motion

For each feature, the model of the relative motion (4.31) is now combined with the constraints (4.32) to obtain a set of linear equations in $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$. This set is then solved in each timestep, yielding the instantaneous motion of the robot.

For each feature, an expression (4.31) holds. In matrix notation and combined for all features, this yields:

$$\begin{bmatrix} \boldsymbol{J}^a_{\boldsymbol{R}} & \boldsymbol{J}^a_{\boldsymbol{F}} & \mathbf{0} & \cdots \\ \boldsymbol{J}^b_{\boldsymbol{R}} & \mathbf{0} & \boldsymbol{J}^b_{\boldsymbol{F}} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \\ \boldsymbol{\tau}^a \\ \boldsymbol{\tau}^b \\ \vdots \end{bmatrix} = \begin{bmatrix} -(\mathbf{t}_u)^a \\ -(\mathbf{t}_u)^b \\ \vdots \end{bmatrix}. \tag{4.36}$$

Formally, this is further reduced by defining matrices $\bar{\boldsymbol{J}}_{\boldsymbol{R}}$, $\bar{\boldsymbol{J}}_{\boldsymbol{F}}$, $\bar{\boldsymbol{T}}_u$ and coordinate vector $\bar{\boldsymbol{\tau}}$:

$$\begin{bmatrix} \bar{\boldsymbol{J}}_{\boldsymbol{R}} & \bar{\boldsymbol{J}}_{\boldsymbol{F}} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \\ \bar{\boldsymbol{\tau}} \end{bmatrix} = \bar{\boldsymbol{T}}_u, \tag{4.37}$$

with

$$\bar{\boldsymbol{J}}_{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{J}^a_{\boldsymbol{R}} \\ \boldsymbol{J}^b_{\boldsymbol{R}} \\ \vdots \end{bmatrix}, \quad \bar{\boldsymbol{J}}_{\boldsymbol{F}} = \begin{bmatrix} \boldsymbol{J}^a_{\boldsymbol{F}} & \mathbf{0} & \cdots \\ \mathbf{0} & \boldsymbol{J}^b_{\boldsymbol{F}} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix},$$

$$\bar{\boldsymbol{T}}_u = \begin{bmatrix} -(\mathbf{t}_u)^a \\ -(\mathbf{t}_u)^b \\ \vdots \end{bmatrix}, \quad \bar{\boldsymbol{\tau}} = \begin{bmatrix} \boldsymbol{\tau}^a \\ \boldsymbol{\tau}^b \\ \vdots \end{bmatrix}.$$

Since $\bar{\boldsymbol{J}}_{\boldsymbol{F}}$ is always of full rank, as it is composed of $\boldsymbol{J}^a_{\boldsymbol{F}}$, $\boldsymbol{J}^b_{\boldsymbol{F}}$,... which are full rank bases, (4.37) leads to an expression for $\bar{\boldsymbol{\tau}}$:

$$\bar{\boldsymbol{\tau}} = \bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1} \left(\bar{\boldsymbol{T}}_u - \bar{\boldsymbol{J}}_{\boldsymbol{R}} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \right). \tag{4.38}$$

For each of the features, a set of constraints (4.32) is defined. In matrix notation, this yields:

$$\begin{bmatrix} \boldsymbol{C}^a_{\boldsymbol{R}} & \boldsymbol{C}^a_{\boldsymbol{F}} & \mathbf{0} & \cdots \\ \boldsymbol{C}^b_{\boldsymbol{R}} & \mathbf{0} & \boldsymbol{C}^b_{\boldsymbol{F}} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \\ \boldsymbol{\tau}^a \\ \boldsymbol{\tau}^b \\ \vdots \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}^a \\ \boldsymbol{u}^b \\ \vdots \end{bmatrix}. \tag{4.39}$$

With the following definitions:

$$\bar{\boldsymbol{C}}_{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{C}^a_{\boldsymbol{R}} \\ \boldsymbol{C}^b_{\boldsymbol{R}} \\ \vdots \end{bmatrix}, \quad \bar{\boldsymbol{C}}_{\boldsymbol{F}} = \begin{bmatrix} \boldsymbol{C}^a_{\boldsymbol{F}} & \mathbf{0} & \cdots \\ \mathbf{0} & \boldsymbol{C}^b_{\boldsymbol{F}} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}, \quad \bar{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{u}^a \\ \boldsymbol{u}^b \\ \vdots \end{bmatrix},$$

(4.39) is reduced to

$$\left[\begin{array}{cc} \bar{\boldsymbol{C}}_{\boldsymbol{R}} & \bar{\boldsymbol{C}}_{\boldsymbol{F}} \end{array}\right] \left[\begin{array}{c} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \\ \bar{\boldsymbol{\tau}} \end{array}\right] = \bar{\boldsymbol{u}}. \tag{4.40}$$

Combined with (4.38), this yields the linear set in $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$:

$$\begin{aligned} \bar{\boldsymbol{C}}_{\boldsymbol{R}}\dot{\boldsymbol{q}}_{\boldsymbol{R}} + \bar{\boldsymbol{C}}_{\boldsymbol{F}}\bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\left(\bar{\boldsymbol{T}}_u - \bar{\boldsymbol{J}}_{\boldsymbol{R}}\dot{\boldsymbol{q}}_{\boldsymbol{R}}\right) &= \bar{\boldsymbol{u}} \\ \Leftrightarrow \left(\bar{\boldsymbol{C}}_{\boldsymbol{R}} - \bar{\boldsymbol{C}}_{\boldsymbol{F}}\bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\bar{\boldsymbol{J}}_{\boldsymbol{R}}\right)\dot{\boldsymbol{q}}_{\boldsymbol{R}} &= \left(\bar{\boldsymbol{u}} - \bar{\boldsymbol{C}}_{\boldsymbol{F}}\bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\bar{\boldsymbol{T}}_u\right), \end{aligned} \tag{4.41}$$

or, with $\boldsymbol{A} = \bar{\boldsymbol{C}}_{\boldsymbol{R}} - \bar{\boldsymbol{C}}_{\boldsymbol{F}}\bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\bar{\boldsymbol{J}}_{\boldsymbol{R}}$ and $\boldsymbol{U} = \bar{\boldsymbol{u}} - \bar{\boldsymbol{C}}_{\boldsymbol{F}}\bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\bar{\boldsymbol{T}}_u$:

$$\boldsymbol{A}\dot{\boldsymbol{q}}_{\boldsymbol{R}} = \boldsymbol{U}. \tag{4.42}$$

According to the rank of $\boldsymbol{A}$, this set of constraints on $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ can be exactly constrained, under- or overconstrained. Each of these cases requires a different approach to solve the set for $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$.

Exactly constrained case

If (4.42) is exactly constrained, the task specification is concluded. There is only one solution $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$, corresponding to:

$$\dot{\boldsymbol{q}}_{\boldsymbol{R}} = \boldsymbol{A}^{-1}\boldsymbol{U}.$$

In every timestep, $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ is calculated and applied to the robot.

Underconstrained case

In the underconstrained case, an infinite number of solutions $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ exists, which realize all constraints. A specific choice needs to be made, to obtain values for $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ to apply to the robot. One possibility is to define a norm and to choose the corresponding minimum norm solution (Doty, Melchiorri, and Bonivento 1993; Klein and Huang 1983; Nakamura 1991). Another possibility is to define extra constraints in the nullspace of $\boldsymbol{A}$ (Liegeois 1977; Aksenov, Voronetskaya, and Fomin 1978). Both cases are briefly discussed below:

(1) Joint space weighting The particular solution, minimizing the weighted joint space norm

$$||\dot{\boldsymbol{q}}_{\boldsymbol{R}}||_{\boldsymbol{W}_q} = \dot{\boldsymbol{q}}_{\boldsymbol{R}}{}^T\boldsymbol{W}_q\dot{\boldsymbol{q}}_{\boldsymbol{R}},$$

is given by the weighted pseudo-inverse:

$$\dot{\boldsymbol{q}}_{\boldsymbol{R}} = \boldsymbol{A}^{\#}\boldsymbol{U}.$$

In the underconstrained case, $\boldsymbol{A}$ is of full row rank and a closed-form expression exists for this pseudo-inverse:

$$\boldsymbol{A}^{\#} = \boldsymbol{W}_q \boldsymbol{A}^T \left[\boldsymbol{A} {\boldsymbol{W}_q}^{-1} \boldsymbol{A}^T \right]^{-1} .$$

In this, the weighting matrix $\boldsymbol{W}_q$ expresses the relative weights of the joint velocities. If the mass matrix of the robot is used as weighting matrix, $||\dot{\boldsymbol{q}}_{\boldsymbol{R}}||_{\boldsymbol{W}_q}$ is proportional to the kinetic energy of the robot. Hence, for that choice the corresponding solution $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ minimizes the instantaneous kinetic energy of the robot.

(2) Nullspace constraints When the robot task is not fully determined, extra degrees of freedom remain available to realize subtasks. Besides the control constraints of the task, the *primary* constraints, the programmer can specify a number of extra, *secondary* constraints, which are not *critical* to the task, but which nevertheless are *desirable*. When the sets of primary and secondary constraints are not conflicting, both sets of constraints are completely realized. When they are conflicting however, the set of primary constraints is fully realized, while the secondary constraints are only realized to such an extent that they don't conflict with the primary constraints.

Consider a set of primary constraints $\boldsymbol{A}\dot{\boldsymbol{q}}_{\boldsymbol{R}} = \boldsymbol{U}$. All solutions realizing the primary constraints are given by:

$$\dot{\boldsymbol{q}}_{\boldsymbol{R}} = \boldsymbol{A}^{\#}\boldsymbol{U} + \left(\boldsymbol{I} - \boldsymbol{A}^{\#}\boldsymbol{A} \right) \dot{\boldsymbol{q}}_{\boldsymbol{sec}}, \tag{4.43}$$

with $\dot{\boldsymbol{q}}_{\boldsymbol{sec}}$ a set of joint velocities. As $\left(\boldsymbol{I} - \boldsymbol{A}^{\#}\boldsymbol{A} \right)$ is a projecting matrix, projecting a vector of joint velocities onto the nullspace of $\boldsymbol{A}$, any vector of joint velocities can be chosen as $\dot{\boldsymbol{q}}_{\boldsymbol{sec}}$.

Now consider a set of secondary constraints $\boldsymbol{B}\dot{\boldsymbol{q}}_{\boldsymbol{R}} = \boldsymbol{V}$. Specific values for $\dot{\boldsymbol{q}}_{\boldsymbol{sec}}$, realizing these secondary constraints, are obtained by solving $\dot{\boldsymbol{q}}_{\boldsymbol{sec}}$ from this set of constraints and (4.43):

$$\begin{aligned} \boldsymbol{B} \left(\boldsymbol{A}^{\#}\boldsymbol{U} + \left(\boldsymbol{I} - \boldsymbol{A}^{\#}\boldsymbol{A} \right) \dot{\boldsymbol{q}}_{\boldsymbol{sec}} \right) &= \boldsymbol{V} \\ \Leftrightarrow \left(\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}^{\#}\boldsymbol{A} \right) \dot{\boldsymbol{q}}_{\boldsymbol{sec}} &= \boldsymbol{V} - \boldsymbol{B}\boldsymbol{A}^{\#}\boldsymbol{U} \\ \Rightarrow \dot{\boldsymbol{q}}_{\boldsymbol{sec}} &= \left(\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}^{\#}\boldsymbol{A} \right)^{\dagger} \left(\boldsymbol{V} - \boldsymbol{B}\boldsymbol{A}^{\#}\boldsymbol{U} \right) . \end{aligned}$$

The corresponding values for $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ are then given by:

$$\dot{\boldsymbol{q}}_{\boldsymbol{R}} = \boldsymbol{A}^{\#}\boldsymbol{U} + \left(\boldsymbol{I} - \boldsymbol{A}^{\#}\boldsymbol{A} \right) \left(\boldsymbol{B} - \boldsymbol{B}\boldsymbol{A}^{\#}\boldsymbol{A} \right)^{\dagger} \left(\boldsymbol{V} - \boldsymbol{B}\boldsymbol{A}^{\#}\boldsymbol{U} \right) . \tag{4.44}$$

In this, the dagger symbol also denotes weighted pseudo-inverse. A different symbol is used here, as both pseudo-inverses in above formulas do not necessarily use the same weighting matrices.

Overconstrained case

In the overconstrained case, the task cannot fully be performed as some of the constraints are conflicting. In this case, the weighted pseudo-inverse yields that solution which minimizes the norm of the constraint violation:

$$||\boldsymbol{A}\dot{\boldsymbol{q}}_{\boldsymbol{R}} - \boldsymbol{U}||_{\boldsymbol{W}_c} = \left(\boldsymbol{A}\dot{\boldsymbol{q}}_{\boldsymbol{R}} - \boldsymbol{U}\right)^T \boldsymbol{W}_c \left(\boldsymbol{A}\dot{\boldsymbol{q}}_{\boldsymbol{R}} - \boldsymbol{U}\right).$$

In the overconstrained case, $\boldsymbol{A}$ is of full column rank and a closed-form expression exists for this pseudo-inverse:

$$\boldsymbol{A}^{\#} = \left[\boldsymbol{A}^T \boldsymbol{W}_c \boldsymbol{A}\right]^{-1} \boldsymbol{A}^T \boldsymbol{W}_c.$$

In this, the weighting matrix $\boldsymbol{W}_c$ expresses the relative weights of errors on the corresponding constraints.

4.4 Conclusions

This chapter presents iTASC, a constraint-based task specification procedure for sensor-based robot tasks, which can deal with more complex tasks than the state of the art (multiple concurrent constraints from different sources and in different frames can be handled), and which can be generally applied to velocity resolved robots with rigid links and joints, and sensor yielding geometric information. The presented specification methodology is a general way to specify tasks according to the Task Function Approach, without explicitly expressing the Task Function and calculating its derivative. To support the task specification, objects and features are introduced and frames are assigned to them. The motion of these frames is then modeled in terms of feature Jacobians and feature twist coordinates. The goal is to choose the frames and feature Jacobians such, that the constraints realizing the task are easily expressible in terms of the feature twist coordinates.

A good choice for the object and feature frames is important, as it allows for easy mathematical expressions for the feature Jacobians. Also the choice for these feature Jacobians is important, as they define the feature twist coordinates, in terms of which the constraints are formulated. A possible point of criticism is that there are no strict rules to choose the object and feature frames, nor to choose the feature Jacobians. Only guidelines are available. This is indeed a disadvantage, as it means that the programmer still needs some insight or experience to specify a task. However, this same disadvantage is also present in other task specification methodologies, notwithstanding their successful application for multiple tasks. For instance, also in the TFF it is difficult to specify a task if the task frame is ill-chosen. This inspired the publication of reference papers such as (Bruyninckx and De Schutter 1996), in which multiple examples are given of tasks, specified according to the TFF. Similarly, this thesis gives multiple examples of tasks specified according to the constraint-based methodology, in Chapter 6. Moreover, the description of the task in terms of a kinematic chain allows to automate part of the methodology. Software tools can generate feature Jacobians from a description of the kinematic chain (for instance, described in a script or modeled in CAD software). In that case, the specific frames in which the Jacobians are expressed is internal to the software, and hence hidden from the person specifying the task. Such specification software is the subject of ongoing research.

A further remark concerning the task specification procedure as described in this chapter, is that all feature Jacobians need to be referred to a common reference point and reference frame in every timestep, to deduce a linear set (4.42). Hence, in every timestep the poses of all object and feature frames need to be known. For simple cases, for example with feature frames which are fixed with respect to their object frames, this involves only straightforward calculations based on the forward position kinematics of the robot. For the minimally invasive surgery example, calculation of these frame poses is more elaborate, as it involves determining the position of the intersection point of the tool (in this case modeled by a line) and the patient (modeled by the tangent plane through the trocar point). As in general these ad hoc methods can become arbitrarily complex, a general way to update the model is needed to render the task specification procedure completely generic. Such a model update procedure is proposed in Section 5.2.

Also, this chapter did not focus on the estimation of geometrical parameters, nor on the DES component. These subjects are discussed in Section 5.2 and Section 5.4, respectively.

Chapter 5

Model Update, Estimation and Discrete Events

5.1 Introduction

Using the basic concepts of the constraint-based task specification methodology as described in the previous chapter, a multitude of tasks can be specified in a generic way. However, the previous chapter does not focus on the following aspects:

- **Model update:** In every timestep, the pose of all object and feature frames must be known. While the programmer can implement an ad-hoc procedure to calculate these frame poses, generic support is desirable as this lessens the work involved when specifying a task. Furthermore, a long term goal is to develop task specification support software which automates as much as possible the task specification process. Model update procedures are a fundamental part of such software.

- **Estimation:** As robots are more and more used in unstructured environments with geometrical uncertainty, sensing and estimation becomes important during the task execution. Sensors gather measurements of the environment, and the differences between the modeled and actual environment are estimated from these measurements. Modeling support to derive the measurement model is desirable.

- **Discrete Event System:** Complex robot tasks usually consist of multiple phases or states with different control constraints needed in every state. A discrete event system is needed to deliberate the need for transitions between different states, with a mechanism which ensures (gradual) transition between those states.

This chapter extends the task specification methodology to cover these three aspects. Section 5.2 discusses the model update procedure. This section assumes that the robot joint positions $\boldsymbol{q_R}$ and the uncontrolled degrees of freedom $\boldsymbol{\chi_U}$ are known. Two procedures are considered: a general procedure, and a procedure which is numerically more stable but which can only be used if the poses of the feature frames can be expressed in terms of a minimal set of coordinates. Section 5.3 discusses how the uncontrolled degrees of freedom $\boldsymbol{\chi_U}$ can be estimated, if these are unknown. Finally, Section 5.2 presents a way to realize smooth crossovers between different states.

5.2 Model Update

To keep the model of the task, that is, the pose of the feature frames, up to date with the actual state of the real world, a model update step is necessary. The model update is concerned with finding the modeled poses of the object and feature frames at time instant $k+1$, given the respective poses at time instant k and the instantaneous motion of the robot. A procedure is proposed here which consists of two steps: a prediction step and a correction step.

5.2.1 General Procedure

The model of the task consists of the object and feature frames for each feature. As the motion of the object frames is determined by the robot motion and possibly also by uncontrolled motion, the poses of these object frames depend on the joint positions and a set of coordinates for the uncontrolled degrees of freedom, $\boldsymbol{\chi_U}$:

$$\begin{aligned} \boldsymbol{T}_w^{o1} &= \boldsymbol{T}_w^{o1}(\boldsymbol{q_R}, \boldsymbol{\chi_U}), \\ \boldsymbol{T}_w^{o2} &= \boldsymbol{T}_w^{o2}(\boldsymbol{q_R}, \boldsymbol{\chi_U}). \end{aligned}$$

The joint positions are known from the robot encoders. The coordinates $\boldsymbol{\chi_U}$ are also assumed to be known, for instance from measurements or from an estimation process (see Section 5.3). Hence, the pose of the object frames is known.

The following procedure, consisting of a prediction and a correction step, now determines the pose of the feature frames. The procedure is based on the first order model of the motion of the feature frames, as expressed by the feature Jacobians.

Prediction

In every timestep, the feature twist coordinates for the motion of each of the feature frames are calculated from the instantaneous joint velocities (equation (4.22)):

$$\bar{\boldsymbol{\tau}} = \bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1} \left(\bar{\boldsymbol{T}}_u + \bar{\boldsymbol{J}}_{\boldsymbol{R}} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \right).$$

For each of the feature frames of feature i, the corresponding twist is found by multiplication of the subvector ${\boldsymbol{\tau}_{I}}^{i}$, ${\boldsymbol{\tau}_{II}}^{i}$ or ${\boldsymbol{\tau}_{III}}^{i}$ from $\bar{\boldsymbol{\tau}}$, with the corresponding feature Jacobian $\boldsymbol{J}^{i}_{\boldsymbol{F}I}$, $\boldsymbol{J}^{i}_{\boldsymbol{F}II}$ or $\boldsymbol{J}^{i}_{\boldsymbol{F}III}$. For instance, for $\mathbf{t}^{f1}_{o1}$:

$$\mathbf{t}^{f1}_{o1} = \boldsymbol{J}_{\boldsymbol{F}I} \boldsymbol{\tau}_{I}.$$

Similar expressions hold for the other feature twists. A prediction for the pose of the feature frames at timestep $k+1$ is then obtained from the pose at timestep k, by integration of the corresponding twist:

$$\widetilde{\boldsymbol{T}}^{f1}_{o1\ k+1} = \exp\left(\mathbf{t}^{f1}_{o1} T_s\right) \widehat{\boldsymbol{T}}^{f1}_{o1\ k}, \tag{5.1}$$

$$\widetilde{\boldsymbol{T}}^{f2}_{f1\ k+1} = \exp\left(\mathbf{t}^{f2}_{f1} T_s\right) \widehat{\boldsymbol{T}}^{f2}_{f1\ k}, \tag{5.2}$$

$$\widetilde{\boldsymbol{T}}^{o2}_{f2\ k+1} = \exp\left(\mathbf{t}^{o2}_{f2} T_s\right) \widehat{\boldsymbol{T}}^{o2}_{f2\ k}. \tag{5.3}$$

In this, $\tilde{}$ denotes a prediction, and $\hat{}$ denotes an estimate. T_s is the timestep. The exponential yields a transformation matrix, as described in Section 3.1.3, page 29.

Correction

Iterative application of (5.1) – (5.3) is prone to accumulation of integration errors due to discretization and due to errors in the geometric model. Therefore the predicted poses of the feature frames need to be corrected. The correction step is based on additional information, contained in the pose closure equations:

$$\boldsymbol{T}^{o1}_{w}(\boldsymbol{q}_{\boldsymbol{R}}, \boldsymbol{\chi}_{\boldsymbol{U}})\ \boldsymbol{T}^{f1}_{o1}\ \boldsymbol{T}^{f2}_{f1}\ \boldsymbol{T}^{o2}_{f2}\ \boldsymbol{T}^{w}_{o2}\ (\boldsymbol{q}_{\boldsymbol{R}}, \boldsymbol{\chi}_{\boldsymbol{U}}) = \boldsymbol{I}_{4\times 4}. \tag{5.4}$$

Errors in the prediction result in imperfect closure of these pose closure equations for each of the features:

$$\boldsymbol{T}^{o1}_{w}(\boldsymbol{q}_{\boldsymbol{R}}, \boldsymbol{\chi}_{\boldsymbol{U}})\ \widetilde{\boldsymbol{T}}^{f1}_{o1}\ \widetilde{\boldsymbol{T}}^{f2}_{f1}\ \widetilde{\boldsymbol{T}}^{o2}_{f2}\ \boldsymbol{T}^{w}_{o2}\ (\boldsymbol{q}_{\boldsymbol{R}}, \boldsymbol{\chi}_{\boldsymbol{U}}) = \boldsymbol{\Delta}. \tag{5.5}$$

In this, $\boldsymbol{\Delta}$ is a homogeneous transformation matrix which deviates from $\boldsymbol{I}_{4\times 4}$.

Linearizing (5.5) and collecting the linearized equations for all features, results in:

$$\bar{\boldsymbol{J}}_{\boldsymbol{F}} \bar{\boldsymbol{\tau}}_{\Delta} + \bar{\boldsymbol{J}}_{\boldsymbol{R}}\ \Delta \boldsymbol{q}_{\boldsymbol{R}} + \bar{\boldsymbol{T}}_{u\Delta} = \bar{\mathbf{t}}_{\Delta}, \tag{5.6}$$

where

$$\bar{\boldsymbol{T}}_{u\Delta} = \begin{bmatrix} (\mathbf{t}_u)^a_\Delta \\ (\mathbf{t}_u)^b_\Delta \\ \vdots \end{bmatrix} \tag{5.7}$$

is a matrix of finite displacements corresponding to deviations of the uncontrolled coordinates $\boldsymbol{\chi_U}$, and $\bar{\mathbf{t}}_\Delta$ is a matrix containing finite displacements corresponding to the homogeneous transformation matrices $\boldsymbol{\Delta}$:

$$\bar{\mathbf{t}}_\Delta = \begin{bmatrix} \mathbf{t}^a_\Delta \\ \mathbf{t}^b_\Delta \\ \cdots \end{bmatrix}. \tag{5.8}$$

For every feature $i = a, b, \ldots$, $\mathbf{t}^i_\Delta$ results from conversion of $\boldsymbol{\Delta}^i$ to an equivalent finite displacement:

$$\mathbf{t}^i_\Delta = \begin{bmatrix} \Delta \boldsymbol{p} \\ \Delta \boldsymbol{\theta} \end{bmatrix},$$

where $\Delta\boldsymbol{p}$ and $\Delta\boldsymbol{\theta}$ follow from:

$$\begin{bmatrix} [\Delta\boldsymbol{\theta}\times] & \Delta\boldsymbol{p} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} = \log(\boldsymbol{\Delta}).$$

The linearization (5.6) corresponds to the twist closure equation (4.31), but with finite displacements $\bar{\boldsymbol{\tau}}_\Delta, \Delta\boldsymbol{q_R}$ and $\bar{\boldsymbol{T}}_{u\Delta}$ instead of time rates $\bar{\boldsymbol{\tau}}, \dot{\boldsymbol{q}}_{\boldsymbol{R}}$ and $\bar{\boldsymbol{T}}_u$. $\bar{\boldsymbol{\tau}}_\Delta$ corresponds to an error on the feature frame poses, $\Delta\boldsymbol{q_R}$ to an error on the robot joint positions and $\bar{\boldsymbol{T}}_{u\Delta}$ to an error on the uncontrolled degrees of freedom. Together, these errors cause the deviation $\bar{\mathbf{t}}_\Delta$, or thus the equivalent matrices $\boldsymbol{\Delta}$.

As there is no physical motion of the robot nor of the uncontrolled degrees of freedom during the correction step, $\Delta\boldsymbol{q_R}$ and $\bar{\boldsymbol{T}}_{u\Delta}$ are zero matrices, and (5.6) reduces to:

$$\begin{aligned} \bar{\boldsymbol{J}}_{\boldsymbol{F}}\bar{\boldsymbol{\tau}}_\Delta &= \bar{\mathbf{t}}_\Delta, \\ \Leftrightarrow \bar{\boldsymbol{\tau}}_\Delta &= \bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\bar{\mathbf{t}}_\Delta. \end{aligned} \tag{5.9}$$

For each feature, the feature frame poses are now corrected by updating the estimate with the corresponding subvector from $\bar{\boldsymbol{\tau}}_\Delta$:

$$\widehat{\boldsymbol{T}}^{f1}_{o1\ k+1} = \exp(\boldsymbol{J_{FI}}\ \boldsymbol{\tau}_{I\Delta})\, \widetilde{\boldsymbol{T}}^{f1}_{o1\ k+1}, \tag{5.10}$$

$$\widehat{\boldsymbol{T}}^{f2}_{f1\ k+1} = \exp(\boldsymbol{J_{FII}}\ \boldsymbol{\tau}_{II\Delta})\, \widetilde{\boldsymbol{T}}^{f2}_{f1\ k+1}, \tag{5.11}$$

$$\widehat{\boldsymbol{T}}^{o2}_{f2\ k+1} = \exp(\boldsymbol{J_{FIII}}\ \boldsymbol{\tau}_{III\Delta})\, \widetilde{\boldsymbol{T}}^{o2}_{f2\ k+1}. \tag{5.12}$$

This yields the feature frame poses at timestep $k+1$.

5.2.2 Minimal Coordinates

In Section 5.2.1, homogeneous transformation matrices are used as a representation for the frame poses. Whereas each homogeneous transformation matrix contains 12 entries, at most six degrees of freedom are present between the feature and object frames. Hence, the representation is *non-minimal*: it has more entries than the number of degrees of freedom it represents. This implies that the entries of the transformation matrices are not independent: mathematical relations exist to which the entries have to comply. However, the integration errors introduced in the model prediction step generally act on all entries of the transformation matrix, hereby possibly disturbing those relations. As the correction step only allows alteration of the feature frames along their respective modeled directions of motion, as given by the feature frame Jacobians $\boldsymbol{J_F}$, this kind of integration errors is not corrected.

This problem does not always occur. For instance, measurements yield information which not necessarily is restricted to the directions of motion allowed in the correction step. Hence, when using such measurements in the model update (Section 5.2.3), more elaborate error correction is possible and the model update becomes more robust. In the case it does occur, the most general solution is to use an ad hoc procedure to calculate the position of the object and feature frames. However, in the case the relative poses of the feature frames can be described by a minimal set of coordinates, a better procedure exists. These coordinates are called the *feature pose coordinates* $\boldsymbol{\chi}$:

$$\boldsymbol{\chi} = \begin{bmatrix} \boldsymbol{\chi}_{I} \\ \boldsymbol{\chi}_{II} \\ \boldsymbol{\chi}_{III} \end{bmatrix},$$

with

$$\begin{aligned} \boldsymbol{T}_{o1}^{f1} &= \boldsymbol{T}_{o1}^{f1}(\boldsymbol{\chi}_{I}), \\ \boldsymbol{T}_{f1}^{f2} &= \boldsymbol{T}_{f1}^{f2}(\boldsymbol{\chi}_{II}), \\ \boldsymbol{T}_{f2}^{o2} &= \boldsymbol{T}_{f2}^{o2}(\boldsymbol{\chi}_{III}). \end{aligned}$$

A possible choice for the feature twist coordinates is then given by the time derivatives of these feature pose coordinates. For instance, consider a task with five degrees of freedom between $f1$ and $o1$:

- $f1$ translates along the X and Y-axis of $o1$, represented by feature pose coordinates χ_1 and χ_2 respectively,
- $f1$ rotates with respect to o1, represented by ZYX-Euler angles χ_3, χ_4 and χ_5.

The choice of the time derivatives $\dot{\chi}_1, \dot{\chi}_2, \ldots, \dot{\chi}_5$ as feature twist coordinates $\boldsymbol{\tau}_I$ yields the following feature Jacobian:

$$\begin{aligned} \mathbf{t}_{o1}^{f1} &= \boldsymbol{J_{FI}}\boldsymbol{\tau}_I \\ &= \begin{bmatrix} 1 & 0 & \\ 0 & 1 & \boldsymbol{0}_{3\times 3} \\ 0 & 0 & \\ 0 & 0 & \\ 0 & 0 & \boldsymbol{E} \\ 0 & 0 & \end{bmatrix} \begin{bmatrix} \dot{\chi}_1 \\ \dot{\chi}_2 \\ \dot{\chi}_3 \\ \dot{\chi}_4 \\ \dot{\chi}_5 \end{bmatrix}. \end{aligned}$$

The prediction-correction steps are now expressed at the level of the feature pose coordinates. The prediction is given by:

$$\widetilde{\boldsymbol{\chi}}_{k+1} = \widehat{\boldsymbol{\chi}}_k + \dot{\boldsymbol{\chi}}_k T_s, \tag{5.13}$$

which yields predicted feature frame poses $\boldsymbol{T}_{o1}^{f1}(\widetilde{\boldsymbol{\chi}}_I)$, $\boldsymbol{T}_{f1}^{f2}(\widetilde{\boldsymbol{\chi}}_{II})$ and $\boldsymbol{T}_{f2}^{o2}(\widetilde{\boldsymbol{\chi}}_{III})$.

The errors on these predicted frames lead to non-closure of the pose closure equations:

$$\boldsymbol{T}_w^{o1}(\boldsymbol{q_R}, \boldsymbol{\chi_U})\ \boldsymbol{T}_{o1}^{f1}(\widetilde{\boldsymbol{\chi}}_I)\ \boldsymbol{T}_{f1}^{f2}(\widetilde{\boldsymbol{\chi}}_{II})\ \boldsymbol{T}_{f2}^{o2}(\widetilde{\boldsymbol{\chi}}_{III})\ \boldsymbol{T}_{o2}^{w}(\boldsymbol{q_R}, \boldsymbol{\chi_U}) = \boldsymbol{\Delta}.$$

In linearized form, this results in:

$$\bar{\boldsymbol{J}}_{\boldsymbol{F}}\ \Delta\boldsymbol{\chi} + \bar{\boldsymbol{J}}_{\boldsymbol{R}}\ \Delta\boldsymbol{q_R} + \bar{\boldsymbol{T}}_{u\Delta} = \bar{\mathbf{t}}_{\Delta}. \tag{5.14}$$

Again, the deviations $\bar{\mathbf{t}}_{\Delta}$ are assumed to be completely caused by an inaccurate position of the predicted feature frame poses. Hence (5.14) reduces to:

$$\begin{aligned} \bar{\boldsymbol{J}}_{\boldsymbol{F}}\ \Delta\bar{\boldsymbol{\chi}} &= \bar{\mathbf{t}}_{\Delta}, \\ \Leftrightarrow \Delta\bar{\boldsymbol{\chi}} &= \bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\bar{\mathbf{t}}_{\Delta}. \end{aligned} \tag{5.15}$$

For each feature, the feature frame poses are now corrected by updating the prediction with the corresponding subvector from $\Delta\bar{\boldsymbol{\chi}}$:

$$\widehat{\boldsymbol{\chi}}_{k+1} = \widetilde{\boldsymbol{\chi}}_{k+1} + \Delta\boldsymbol{\chi},$$

yielding the estimate $\widehat{\boldsymbol{\chi}}_{k+1}$. From this estimate, the frame poses are recalculated: $\boldsymbol{T}_{o1}^{f1}(\widehat{\boldsymbol{\chi}}_I)$, $\boldsymbol{T}_{f1}^{f2}(\widehat{\boldsymbol{\chi}}_{II})$ and $\boldsymbol{T}_{f2}^{o2}(\widehat{\boldsymbol{\chi}}_{III})$. As these poses are always calculated from the minimal set of coordinates, consistency of the pose representation is guaranteed.

5.2.3 Measurements

In sensor-based robot tasks, sensor measurements are mostly used in the control loop. However, measurements can also provide geometrical information about the robot and its environment, and hence are useful in the model update.

As described in Section 4.3.3, for geometrical sensors such as a camera or laser distance sensor, the measurements are function of the relative pose of the objects. Hence, in general, the measurement equation is given by:

$$z = g(\boldsymbol{d}_{o1}^{o2}). \tag{5.16}$$

The measurement equation for dynamical sensors such as a force/torque sensor is also given by (5.16), if the assumption of quasistatic behavior is valid. Time derivation of (5.16) yields:

$$\begin{aligned} \dot{z} &= \frac{dg}{d\boldsymbol{d}_{o1}^{o2}}\dot{\boldsymbol{d}}_{o1}^{o2} \\ &= \frac{dg}{d\boldsymbol{d}_{o1}^{o2}}\boldsymbol{E}^{-1}\mathbf{t}_{o1}^{o2}. \end{aligned} \tag{5.17}$$

Using the twist closure equations (4.31), $\mathbf{t}_{o1}^{o2}$ can always be written as a linear combination of $\dot{\boldsymbol{q}}_{\boldsymbol{R}}, \boldsymbol{\tau}$ and $\mathbf{t}_u$. Introducing this result in (5.17) reveals that $\dot{z}$ can always be written as a linear combination of $\dot{\boldsymbol{q}}_{\boldsymbol{R}}, \boldsymbol{\tau}$ and $\mathbf{t}_u$:

$$\dot{z} = \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{R}} & \boldsymbol{H}_{\boldsymbol{F}} & \boldsymbol{H}_{\boldsymbol{U}} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \\ \boldsymbol{\tau} \\ \mathbf{t}_u \end{bmatrix}. \tag{5.18}$$

The matrices $\boldsymbol{H}_{\boldsymbol{R}}, \boldsymbol{H}_{\boldsymbol{F}}$ and $\boldsymbol{H}_{\boldsymbol{U}}$ express the rate of the sensor measurement for respectively a unit joint velocity, feature frame twist and uncontrolled twist. Collecting all the measurements in a vector $\boldsymbol{z}$ leads to the general expression:

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \bar{\boldsymbol{H}}_{\boldsymbol{R}} & \bar{\boldsymbol{H}}_{\boldsymbol{F}} & \bar{\boldsymbol{H}}_{\boldsymbol{U}} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \\ \bar{\boldsymbol{\tau}} \\ \bar{\boldsymbol{T}}_u \end{bmatrix}. \tag{5.19}$$

According to (5.16), a prediction for $\boldsymbol{z}$ is calculated, based on the modeled frame positions:

$$\widetilde{\boldsymbol{z}} = \boldsymbol{g}(\widetilde{\boldsymbol{d}}_{o1}^{o2}). \tag{5.20}$$

In this, $\widetilde{\boldsymbol{d}}_{o1}^{o2}$ is the predicted finite displacement between $o1$ and $o2$, equivalent to the homogeneous transformation matrix

$$\widetilde{\boldsymbol{T}}_{o1}^{o2} = \widetilde{\boldsymbol{T}}_{o1}^{f1}\,\widetilde{\boldsymbol{T}}_{f1}^{f2}\,\widetilde{\boldsymbol{T}}_{f2}^{o2}.$$

(5.19) corresponds to the linearization of (5.16), and relates small deviations $\Delta\boldsymbol{z} = (\widetilde{\boldsymbol{z}} - \boldsymbol{z})$ to variations $\Delta\boldsymbol{q_R}, \bar{\boldsymbol{\tau}}_\Delta$ and $\bar{\boldsymbol{T}}_{u\Delta}$:

$$\Delta\boldsymbol{z} = \begin{bmatrix} \bar{\boldsymbol{H}}_{\boldsymbol{R}} & \bar{\boldsymbol{H}}_{\boldsymbol{F}} & \bar{\boldsymbol{H}}_{\boldsymbol{U}} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{q_R} \\ \bar{\boldsymbol{\tau}}_\Delta \\ \bar{\boldsymbol{T}}_{u\Delta} \end{bmatrix}. \tag{5.21}$$

Assuming again that only the feature frame poses cause inaccuracies, yields:

$$\begin{aligned} \Delta\boldsymbol{z} &= \bar{\boldsymbol{H}}_{\boldsymbol{F}} \bar{\boldsymbol{\tau}}_\Delta, \\ \Rightarrow \bar{\boldsymbol{\tau}}_\Delta &= \bar{\boldsymbol{H}}_{\boldsymbol{F}}^{\#} \Delta\boldsymbol{z}. \end{aligned} \tag{5.22}$$

The information of the pose closure and the measurements can be combined in a single correction step:

$$\bar{\boldsymbol{\tau}}_\Delta = \begin{bmatrix} \boldsymbol{J_F} \\ \bar{\boldsymbol{H}}_{\boldsymbol{F}} \end{bmatrix}_{\boldsymbol{W}}^{\#} \begin{bmatrix} \bar{\mathbf{t}}_\Delta \\ \Delta\boldsymbol{z} \end{bmatrix}. \tag{5.23}$$

In this, the weights $\boldsymbol{W}$ express the deliberation between the information in the pose closure equations on the one hand, and in the sensor measurements on the other hand. Usually, high weights are chosen for the pose closure equations and lower weights for the measurements, reflecting the intrinsic correctness of the pose closure equations, and the noise on the measurements.

5.3 Estimation

One of the major reasons to incorporate sensors in a robotic setup, is to gain information about geometrical uncertainty in the robot environment. One possibility is to use the sensor measurements directly in the control loop. This is for instance the case in compliant motion tasks, where a force control loop ensures a desired contact force between objects. Another possibility is to use estimation techniques to update a geometric model of the robot and its environment according to the measurements.

This section extends the model update procedure to include estimation of uncertain coordinates, which represent uncontrolled degrees of freedom and affect the pose of the object frames.

5.3.1 Modeling of the Uncertain Degrees of Freedom

In the previous section, coordinates $\boldsymbol{\chi_U}$ were introduced, to model uncontrolled degrees of freedom. Also unknown parameters can be represented by these coordinates, to estimate them based on sensor measurements. The pose of the object frames is given by:

$$\boldsymbol{T}_w^{o1} = \boldsymbol{T}_w^{o1}(\boldsymbol{q_R}, \boldsymbol{\chi_U}), \tag{5.24}$$

$$\boldsymbol{T}_w^{o2} = \boldsymbol{T}_w^{o2}(\boldsymbol{q_R}, \boldsymbol{\chi_U}). \tag{5.25}$$

For reasons of notational simplicity, the discussion here is limited to the estimation of integrable coordinates $\boldsymbol{\chi}$ and $\boldsymbol{\chi_U}$. However, estimation based on differential coordinates $\boldsymbol{\tau}$ and $\boldsymbol{\tau_U}$ is equally possible. The extra coordinates represent extra degrees of freedom in the model. These allow for adaption of the model along other directions than those defined by $\boldsymbol{J_F}$ in the model of motion of the feature frames. The motion of the object frames according to variations of these coordinates $\boldsymbol{\chi_U}$ is also described:

$$\mathbf{t}_{o2}^{o1} = \boldsymbol{J_R}\dot{\boldsymbol{q}}_{\boldsymbol{R}} + \boldsymbol{J_U}\dot{\boldsymbol{\chi}}_{\boldsymbol{U}}. \tag{5.26}$$

Describing this motion is similar to the modeling of the motion of the feature frames. Consider for instance the following task. A robot needs to locate a barrel, in order to take a sample of its contents. The drum stands on the floor, so its height is known. However, the position in on the floor, described by two coordinates x and y, as well as the rotation of the drum around the vertical axis, ϕ, are unknown. Hence:

$$\boldsymbol{T}_{o1}^{o2} = \boldsymbol{T}_{o1}^{w}(\boldsymbol{q_R})\, \boldsymbol{T}_w^{o2}(\boldsymbol{\chi_U}),$$

with

$$\boldsymbol{\chi_U} = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix}.$$

The motion of the object frame according to changes of the uncertain coordinates is given by:

$$\begin{aligned} {}_w\mathbf{t}_w^{o2} &= \boldsymbol{J_U}\dot{\boldsymbol{\chi}}_{\boldsymbol{U}} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix}. \end{aligned}$$

5.3.2 Integration of Estimation in the Model Update

The prediction and correction steps of the model update are now extended to include estimation of the uncertain coordinates.

Prediction

In general, the system model, relating the values for $\boldsymbol{\chi_U}$ at timestep $k+1$ to those at timestep k, is given by:

$$\boldsymbol{\chi_U}_{k+1} = \boldsymbol{f}(\boldsymbol{\chi_U}_k). \tag{5.27}$$

For instance, for the previous example, in which a barrel stands on a floor at an unknown position, the coordinates $\boldsymbol{\chi_U}$ are constant, or:

$$\boldsymbol{\chi_U}_{k+1} = \boldsymbol{\chi_U}_k.$$

In another setting, where the barrels are supplied on a conveyor belt, a constant velocity model is more appropriate:

$$\boldsymbol{\chi_U}_{k+1} = \boldsymbol{\chi_U}_k + \dot{\boldsymbol{\chi}}_{\boldsymbol{U}} T_s.$$

The system model yields a prediction for $\boldsymbol{\chi_U}$:

$$\widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}k+1} = \boldsymbol{f}(\widehat{\boldsymbol{\chi}}_{\boldsymbol{U}k}).$$

This prediction can be combined with (5.13) into one prediction vector:

$$\begin{bmatrix} \widetilde{\boldsymbol{\chi}} \\ \widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}} \end{bmatrix}_{k+1}.$$

A prediction of the feature frame poses then follows from $\widetilde{\boldsymbol{\chi}}$, as explained in Section 5.2.2. On the other hand, a prediction of the poses of $o1$ and $o2$ with respect to the world reference frame follows by substituting $\widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}k+1}$ in (5.24) and (5.25). Geometrical errors in these predicted frames result in non-closure of the pose closure equations for each feature:

$$\widetilde{\boldsymbol{T}}_w^{o1}(\boldsymbol{q_R}, \widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}})\ \widetilde{\boldsymbol{T}}_{o1}^{f1}\ \widetilde{\boldsymbol{T}}_{f1}^{f2}\ \widetilde{\boldsymbol{T}}_{f2}^{o2}\ \widetilde{\boldsymbol{T}}_{o2}^{w}\ (\boldsymbol{q_R}, \widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}}) = \boldsymbol{\Delta}. \tag{5.28}$$

Correction

The twist closure equation is given by:

$$\mathbf{t}_w^{o1} + \mathbf{t}_{o1}^{f1} + \mathbf{t}_{f1}^{f2} + \mathbf{t}_{f2}^{o2} + \mathbf{t}_{o2}^{w} = \mathbf{0}.$$

Using $\mathbf{t}_{o2}^{o1} = \boldsymbol{J_R}\dot{\boldsymbol{q}}_{\boldsymbol{R}} + \boldsymbol{J_U}\dot{\boldsymbol{\chi}}_{\boldsymbol{U}}$ and $\mathbf{t}_{o1}^{o2} = \boldsymbol{J_F}\dot{\boldsymbol{\chi}}$, this results in:

$$\boldsymbol{J_R}\dot{\boldsymbol{q}}_{\boldsymbol{R}} + \boldsymbol{J_F}\dot{\boldsymbol{\chi}} + \boldsymbol{J_U}\dot{\boldsymbol{\chi}}_{\boldsymbol{U}} = \mathbf{0}.$$

Similar to (5.6), this twist closure equation corresponds to the first order linearization of the pose closure equation, and hence:

$$\begin{aligned} \boldsymbol{J_R}\Delta\boldsymbol{q_R} + \boldsymbol{J_F}\Delta\boldsymbol{\chi} + \boldsymbol{J_U}\Delta\boldsymbol{\chi_U} &= \mathbf{t}_\Delta, \\ \Leftrightarrow \begin{bmatrix} \boldsymbol{J_R} & \boldsymbol{J_F} & \boldsymbol{J_U} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{q_R} \\ \Delta\boldsymbol{\chi} \\ \Delta\boldsymbol{\chi_U} \end{bmatrix} &= \mathbf{t}_\Delta. \end{aligned} \tag{5.29}$$

The time rate of the measurement equation is also expressed in terms of the coordinates $\boldsymbol{\chi_U}$:

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \boldsymbol{H_R} & \boldsymbol{H_F} & \boldsymbol{H_U} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \\ \dot{\boldsymbol{\chi}} \\ \dot{\boldsymbol{\chi}}_{\boldsymbol{U}} \end{bmatrix}. \tag{5.30}$$

For small deviations $\Delta \boldsymbol{z} = (\widetilde{\boldsymbol{z}} - \boldsymbol{z})$, (5.30) expresses the first order linearization:

$$\Delta \boldsymbol{z} = \begin{bmatrix} \boldsymbol{H_R} & \boldsymbol{H_F} & \boldsymbol{H_U} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{q_R} \\ \Delta \boldsymbol{\chi} \\ \Delta \boldsymbol{\chi_U} \end{bmatrix}. \tag{5.31}$$

Combined with (41), and assuming no error on the robot joint positions, or $\Delta \boldsymbol{q_R} = 0$, (5.31) yields:

$$\begin{bmatrix} \boldsymbol{J_F} & \boldsymbol{J_U} \\ \boldsymbol{H_F} & \boldsymbol{H_U} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{\chi} \\ \Delta \boldsymbol{\chi_U} \end{bmatrix} = \begin{bmatrix} \mathbf{t}_\Delta \\ \Delta \boldsymbol{z} \end{bmatrix}. \tag{5.32}$$

From this, the predictions are corrected:

$$\begin{bmatrix} \widehat{\boldsymbol{\chi}} \\ \widehat{\boldsymbol{\chi}}_{\boldsymbol{U}} \end{bmatrix}_{k+1} = \begin{bmatrix} \widetilde{\boldsymbol{\chi}} \\ \widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}} \end{bmatrix}_{k+1} + \begin{bmatrix} \boldsymbol{J_F} & \boldsymbol{J_U} \\ \boldsymbol{H_F} & \boldsymbol{H_U} \end{bmatrix}^{\#}_{\boldsymbol{W}} \begin{bmatrix} \mathbf{t}_\Delta \\ \Delta \boldsymbol{z} \end{bmatrix}. \tag{5.33}$$

$\boldsymbol{W}$ is again a weighting matrix which determines the relative importance of the pose closure equation and the measurements in the correction.

5.3.3 Kalman Filter Estimation

Another approach to the model update and estimation procedure, using a Kalman Filter, is described in this section. The parameters defining the pose of the robot, the object and the feature frames, are $\boldsymbol{q_R}, \boldsymbol{\chi}$ and $\boldsymbol{\chi_U}$. As no uncertainty is assumed on $\boldsymbol{q_R}$, the state $\boldsymbol{x}$ of the Kalman Filter is given by:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\chi} \\ \boldsymbol{\chi_U} \end{bmatrix}.$$

The Kalman Filter approach also consist of a prediction and a correction step.

Prediction

The Kalman prediction step is based on the general model update procedure. Based on the system equations (5.13) and (5.27), a prediction for $\boldsymbol{\chi}$ and $\boldsymbol{\chi_U}$ is calculated. Then, the values of $\boldsymbol{\chi}$ are corrected as described in Section 5.2.1, so the pose loop closure equations hold. The linearized system matrix for this step is obtained from the system equations, and the state covariance $\boldsymbol{P}$ adapted according to the Kalman Filter equations.

Correction

In the Kalman correction step, the predictions are corrected according to sensor measurement. The measurement equation, relating the state to the measurements, is given by:

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{\chi}, \boldsymbol{\chi_U}). \tag{5.34}$$

According to (5.31), the linearized form of this measurement equation is given by:

$$\Delta \boldsymbol{z} = \begin{bmatrix} \boldsymbol{H_F} & \boldsymbol{H_U} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{\chi} \\ \Delta \boldsymbol{\chi_U} \end{bmatrix}, \tag{5.35}$$

if no error on the joint positions is assumed ($\Delta \boldsymbol{q_R} = 0$). The predicted measurement $\widetilde{\boldsymbol{z}}$ is given by:

$$\widetilde{\boldsymbol{z}} = \boldsymbol{h}(\widetilde{\boldsymbol{\chi}}, \widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}}), \tag{5.36}$$

while the real measurement is obtained from the sensor.

After the Kalman correction, the pose closure equations should still hold. To this end, the pose closure equations are used as extra measurement equations. The none-closure of the pose loops is not a real measurement, in the sense that they do not provide *external* information to the estimator. The pose closure equations are *constraints* between the state variables $\boldsymbol{\chi}$ and $\boldsymbol{\chi_U}$. The linearization of the pose loops is given, according to (41), by:

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{J_F} & \boldsymbol{J_U} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{\chi} \\ \Delta \boldsymbol{\chi_U} \end{bmatrix}, \tag{5.37}$$

with predicted measurement $-\mathbf{t}_\Delta = \log(\Delta)$ and as real measurement $\mathbf{0}$, which expresses closure of the loop.

Combined with (5.35), this yields the linearized Kalman Filter measurement equation $\boldsymbol{z} = \boldsymbol{H}\boldsymbol{x}$, with:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{J_F} & \boldsymbol{J_U} \\ \boldsymbol{H_F} & \boldsymbol{H_U} \end{bmatrix}. \tag{5.38}$$

The measurement covariance $\boldsymbol{R}$ expresses the uncertainty on every measurement. Often, the measurements are independent, and a diagonal covariance matrix can be used. For the loop closure equations, the measurement covariance is in theory zero: in reality, the pose loop is always closed. However, the measurement model is linearized, which introduces variance. Hence, adding extra measurement noise is a way to cope with this variance.

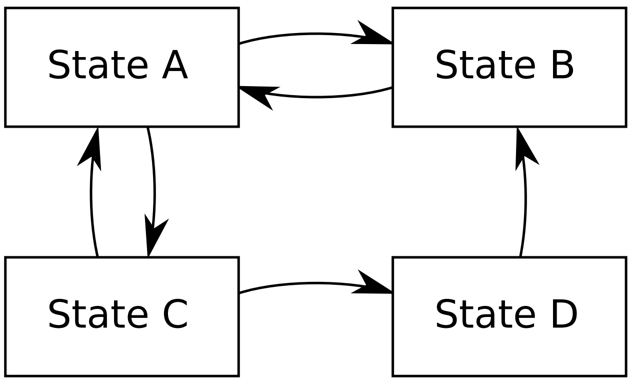


FIGURE 5.1: An example of a state diagram. The arrows denote possible transitions between states.

5.4 Discrete Event System

In true robotic applications, different states are present. For instance, for a compliant motion task, the different states could be: *approach, make contact, move in contact, retract,* and *error*. Figure 5.1 shows an example of a possible state diagram.

In each of the states, different control constraints are active. Consider two states, A and B, with their corresponding set of constraints, and weight matrices $\boldsymbol{W}_A$ and $\boldsymbol{W}_B$. When an abrupt state change is needed from A to B, the constraints of state A are replaced by those of state B. To ensure a smooth transition between the states however, both sets of constraints are activated, with weighting matrices $\boldsymbol{W}^*_A$ and $\boldsymbol{W}^*_B$, according to:

$$\begin{aligned}\boldsymbol{W}^*_A &= \lambda_A \boldsymbol{W}_A,\\ \boldsymbol{W}^*_B &= \lambda_B \boldsymbol{W}_B.\end{aligned}$$

λ_A and λ_B are scale factors between one and zero. During the transition, λ_A is smoothly changed from one to zero, while λ_B is changed from zero to one. Figure 5.2 shows an example, where λ_A and λ_B are changed in a linear way.

5.5 Conclusions

This chapter focuses on a model update procedure, the estimation of unknown geometric parameters and the discrete event system. The model update is concerned with keeping the model, that is, the pose of the object and feature frames, up to date with the actual state of the real world. Based on the known

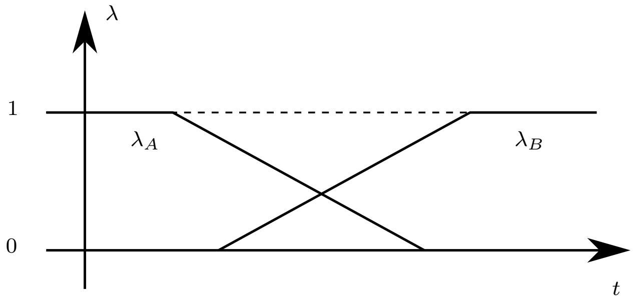


Figure 5.2: An example of linear change of the weighting matrix scale factors lambda, to ensure smooth transition between two states.

joint positions $\boldsymbol{q_R}$, uncontrolled coordinates $\boldsymbol{\chi_U}$ and feature frame poses at time instant k, the feature frame poses at timestep $k+1$ are calculated.

To estimate geometric parameters, these parameters are modeled as uncontrolled (and in this case also unknown) coordinates $\boldsymbol{\chi_U}$. The motion degrees of freedom of these uncontrolled coordinates are modeled in terms of a Jacobian $\boldsymbol{J_U}$, similar to the feature Jacobian $\boldsymbol{J_F}$. Similarly to the model update procedure, the coordinates $\boldsymbol{\chi_U}$ are then estimated in a prediction and a correction step.

Finally, a procedure is presented to make a smooth transition between different states of a robot task.

Chapter 6

Applications

6.1 Introduction

In this chapter, several example applications are discussed, each of which illustrate different aspects of the task specification formalism. The first example application (Section 6.2) is based on the minimally invasive surgery example of Section 4.2, and shows how an existing task can easily be extended to a new setting with extra DOF, by changing parts of the model and adding constraints. The second and third application are the other example applications of Chapter 4: the forming task (Section 6.3) and the inspection task (Section 6.4). The former is an example of a task involving force control and contact between curved surfaces, and the latter task concerns a highly redundant robot and features transitions between multiple states, in each of which different control constraints are active. The fourth application is a common example of a task which cannot be easily expressed in the TFF: the incompatible seam following task (Section 6.5). The final two applications are laser tracing, involving underconstrained specification and estimation of geometric parameters (Section 6.6), and human-robot shared control, involving overconstrained specification, realizing shared control between a human operator, the robot and visual servoing (Section 6.7).

6.2 8 DOF Minimally Invasive Surgery

In Section 4.2 (page 38), a minimally invasive surgery task was discussed as an introduction to the task specification formalism. In that task, a six DOF robot inserts a laparascopic tool into a patient's body through a hole, called the trocar. The tool has a gripper at its end point. The robot's motion is controlled such that the trocar point stays at its desired position, and the

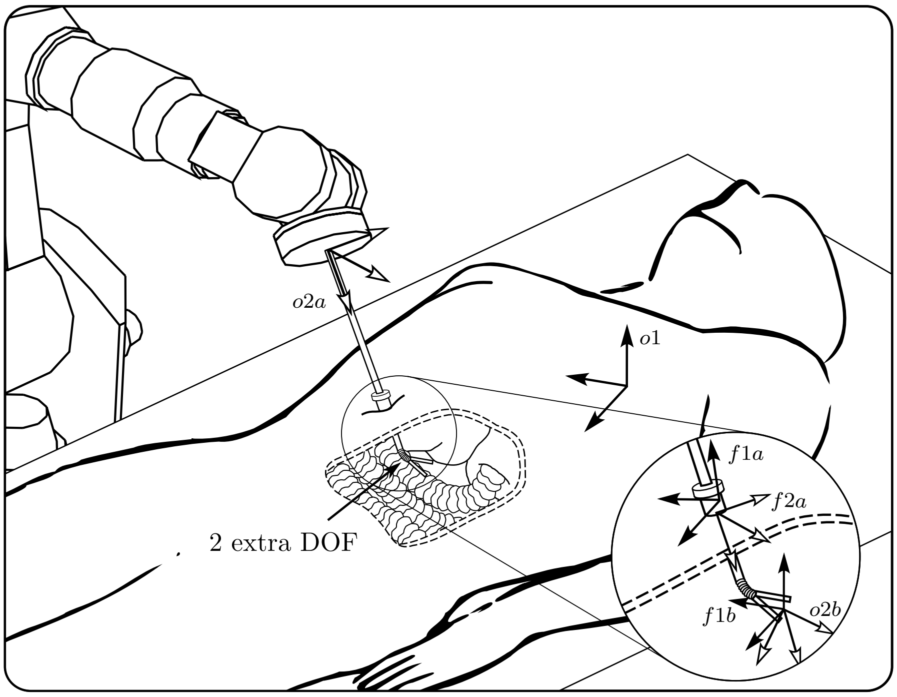


Figure 6.1: The object and feature frames for a minimally invasive surgery task, with 8 DOF (6 DOF of the robot, and 2 extra DOF of the tool). *f*2*b* coincides with *o*2*b*.

endpoint of the tool moves along a specified trajectory to reach an organ in the patient's body.

The application described in this section starts from the same setting but uses a different laparascopic tool. The tool has two extra rotational DOF at its tip (Figure 6.1). This allows complete 6D positioning of the gripper at the end of the tool with respect to an organ in the body. Very little changes to the steps from Section 4.2 are needed to incorporate the extra DOF into the task.

As with the initial task of Section 4.2, the new setting also suggests two motion constraints, and hence two features: one regarding the position of the trocar point (feature *a*), and one regarding the motion of the endpoint of the tool (feature *b*). With respect to theobject and feature frames and the feature Jacobians for feature *a*, nothing changes. Hence, these frames are chosen as follows:

- Frame *o*1 is fixed at a reference position on the patient's body.

- Frame $o2a$ is fixed to the mounting plate of the robot. It has its origin at the attachment point of the laparascopic tool, and it is oriented with its Z-axis along the tool.

- Frame $f1a$ is rigidly attached to $o1$, with its origin at the *desired* position for the trocar point, and its Z-axis normal to the patient's body.

- Frame $f2a$ is located on the laparascopic tool, with its origin at the *actual* trocar point, while its orientation is the same as that of $o2a$.

Also the feature Jacobians for feature a are the same as in Section 4.2:

$$\begin{aligned} \mathbf{t}_{o1}^{f1a} &= \boldsymbol{J}_{\boldsymbol{FI}}^{a} \boldsymbol{\tau_{I}}^{a}, \\ &= \mathbf{0}, \end{aligned} \tag{6.1}$$

$$\begin{aligned} {}^{f2a}_{f1a}\mathbf{t}_{f1a}^{f2a} &= \boldsymbol{J}_{\boldsymbol{FII}}^{a} \boldsymbol{\tau_{II}}^{a}, \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_1^a \\ \tau_2^a \\ \tau_3^a \\ \tau_4^a \\ \tau_5^a \end{bmatrix}, \end{aligned} \tag{6.2}$$

$$\begin{aligned} {}^{f2a}_{o2a}\mathbf{t}_{o2a}^{f2a} &= \boldsymbol{J}_{\boldsymbol{FIII}}^{a} \boldsymbol{\tau_{III}}^{a}, \\ &= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \tau_6^a \end{bmatrix}. \end{aligned} \tag{6.3}$$

For feature b, there are some changes in the assignment of the frames to reflect the two extra DOF at the tip of the tool. These DOF are extra joints of the robot and influence the motion of the gripper. Hence a new object frame, $o2b$, is introduced at the center of the gripper. The $f2b$ frame is chosen to coincide with $o2b$. The rest of the frames is unaltered: $f1b$ has its origin at the same point as the origin of $f2b$, but the same orientation as $o1$. This leads to the following feature Jacobians:

$$\begin{aligned} {}^{f1b}_{o1}\mathbf{t}^{f1b}_{o1} &= \boldsymbol{J}^{b}_{\boldsymbol{FI}}\boldsymbol{\tau_{I}}^{b}, \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tau^{b}_{1} \\ \tau^{b}_{2} \\ \tau^{b}_{3} \end{bmatrix}, \end{aligned} \tag{6.4}$$

$$\begin{aligned} {}^{f2b}_{f1b}\mathbf{t}^{f2b}_{f1b} &= \boldsymbol{J}^{b}_{\boldsymbol{FII}}\boldsymbol{\tau_{II}}^{b}, \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau^{b}_{4} \\ \tau^{b}_{5} \\ \tau^{b}_{6} \end{bmatrix}, \end{aligned} \tag{6.5}$$

$$\begin{aligned} \mathbf{t}^{f2b}_{o2b} &= \boldsymbol{J}^{b}_{\boldsymbol{FIII}}\boldsymbol{\tau_{III}}^{b}, \\ &= \mathbf{0}. \end{aligned} \tag{6.6}$$

Note that these feature Jacobians are the same as the ones in the setting of Section 4.2.

The twists of $o2a$ and $o2b$ are given by:

$$ {}_{w}\mathbf{t}^{o2a}_{w} = \boldsymbol{J}_{\boldsymbol{R}1}\dot{\boldsymbol{q}}_{\boldsymbol{R}}, \tag{6.7}$$

$$ {}_{w}\mathbf{t}^{o2b}_{w} = \boldsymbol{J}_{\boldsymbol{R}2}\dot{\boldsymbol{q}}_{\boldsymbol{R}}. \tag{6.8}$$

$\boldsymbol{J}_{\boldsymbol{R}2}$ is the full Jacobian of the robot and the tool. $\boldsymbol{J}_{\boldsymbol{R}1}$ is the Jacobian to the sixth link. $\boldsymbol{J}_{\boldsymbol{R}1}$ and $\boldsymbol{J}_{\boldsymbol{R}2}$ are the same matrices, except for the last two columns of $\boldsymbol{J}_{\boldsymbol{R}1}$ which contain only zeros:

$$\boldsymbol{J}_{\boldsymbol{R}1} = \boldsymbol{J}_{\boldsymbol{R}2} \begin{bmatrix} \boldsymbol{I}_{6\times 6} & \boldsymbol{0}_{6\times 2} \\ \boldsymbol{0}_{2\times 6} & \boldsymbol{0}_{2\times 2} \end{bmatrix}. \tag{6.9}$$

Also the constraints are unaltered with respect to the previous setting. However, three constraints are added to also impose the relative orientation of $f1b$ and $f2b$:

- To realize the desired position of the trocar two constraints are needed, imposing the values of τ^{a}_{1} and τ^{a}_{2} to the outcome of a position controller. This leads to the following set of constraints:

$$\tau^{a}_{1} = k_{fb}(x^{a}_{desired} - x^{a}_{actual}), \tag{6.10}$$

$$\tau^{a}_{2} = k_{fb}(y^{a}_{desired} - y^{a}_{actual}). \tag{6.11}$$

In this, k_{fb} is a feedback constant, x^a_{actual} and y^a_{actual} are the coordinates of the origin of $f2a$, expressed in $f1a$, and $x^a_{desired}$ and $y^a_{desired}$ the desired values for these coordinates[1].

- Three further position constraints are needed to impose the translational motion of the endpoint of the tool. These are similar to the previous constraints, but now concern τ^b_1, τ^b_2 and τ^b_3, as these coordinates express the X, Y and Z translational velocities of $f1b$ with respect to $o1$, or thus of the endpoint of the tool with respect to the patient's body:

$$\begin{aligned} \tau^b_1 &= k_{fb}(x^b_{desired} - x^b_{actual}), && (6.12)\\ \tau^b_2 &= k_{fb}(y^b_{desired} - y^b_{actual}), && (6.13)\\ \tau^b_3 &= k_{fb}(z^b_{desired} - z^b_{actual}). && (6.14)\\ & && (6.15) \end{aligned}$$

  In this, x^b_{actual}, y^b_{actual} and z^b_{actual} correspond to the x, y and z-coordinates of $f1b$, expressed in $o1$, while $x^b_{desired}$, $y^b_{desired}$ and $z^b_{desired}$ correspond to the desired values for these coordinates.

- Finally, another three position constraints are needed to impose the rotational motion of the endpoint of the tool. These concern τ^b_4, τ^b_5 and τ^b_6, as these coordinates express the X, Y and Z rotational velocities of $f1b$ with respect to $o1$. A possible choice to express the constraints is in terms of the desired Euler angles. This requires the introduction of an integrating factor $\boldsymbol{E}$ to relate the rotational velocities to Euler angle rates:

$$\boldsymbol{E}^{-1}\begin{bmatrix} \tau^b_4 \\ \tau^b_5 \\ \tau^b_6 \end{bmatrix} = \begin{bmatrix} k_{fb}(\phi^b_{desired} - \phi^b_{actual}) \\ k_{fb}(\theta^b_{desired} - \theta^b_{actual}) \\ k_{fb}(\psi^b_{desired} - \psi^b_{actual}) \end{bmatrix}. \qquad (6.16)$$

  In this, ϕ^b_{actual}, θ^b_{actual} and ψ^b_{actual} correspond to the ZYX-Euler angles between $f2b$ and $f1b$, while $\phi^b_{desired}$, $\theta^b_{desired}$ and $\psi^b_{desired}$ correspond to the desired values for these angles.

6.3 Forming Task

This section deals with the forming task introduced in Section 4.3.1 (Figure 6.2). In the task, a plate is placed between two spheres by a peripheral mechanism in the robot workcell. The robot then rolls one of the spheres over

[1] As previously stated, it is possible to use different feedback constants or different types of controllers. For the simplicity of notation however, always proportional controllers are used, with the same feedback constant k_{fb}.

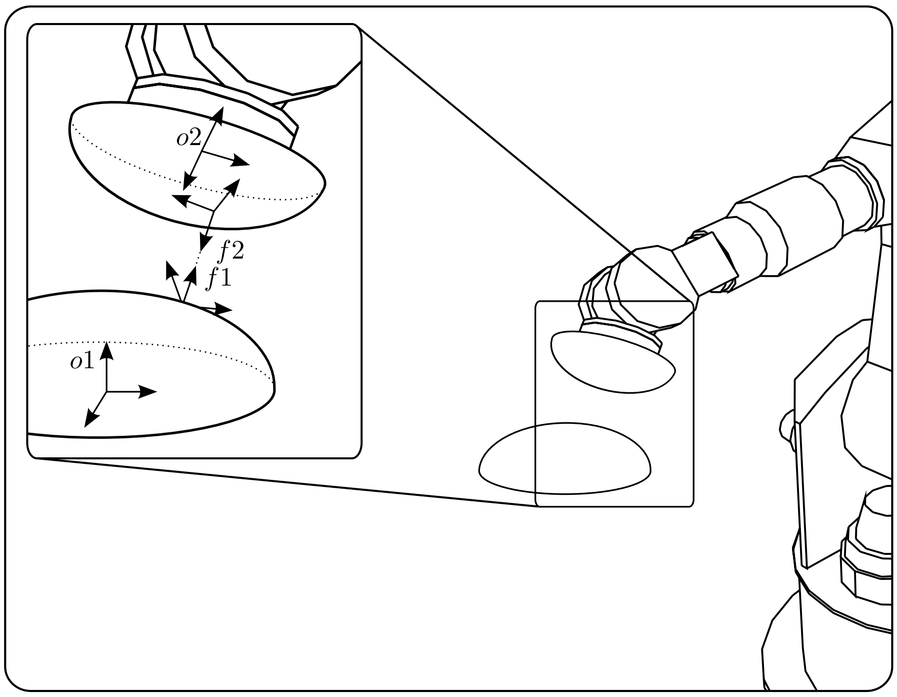


Figure 6.2: The object and feature frames for a forming task. In the task, a plate (not shown) is placed between two spheres, by a peripheral mechanism in the robot workcell. The robot then rolls one of the spheres over the other while maintaining a contact force. This deforms the plate into a curved surface.

the other while maintaining a contact force. This deforms the plate into a curved surface.

As the task consists of relative motion of the spheres, these are the relevant objects. Furthermore, the task is determined by the motion of the contact points on each of the spheres' surfaces, so these contact points form the relevant feature. However, the task also includes states where there is no contact, for instance when loading or unloading a plate. In order to deal with all states in the same way, the notion of 'contact point' is extended to *that point on the sphere's surface, which is the closest to the other sphere*. In other words, when there is no contact, the points on the spheres' surfaces which correspond to the shortest distance between the spheres are chosen instead of actual contact points.

The object and feature frames are now chosen as follows (Figure 6.2):

- $o1$ is placed at the center of the fixed sphere. It has its Z-axis perpendicular to the base of the sphere.

- $o2$ is placed at the center of the sphere which is moved by the robot, also with Z-axis perpendicular to the base of the sphere.

- $f1$ has its origin at that point of the fixed sphere which is the nearest to the other sphere. It has its X-axis along the normal of the sphere, and its Y and Z-axes such, that its orientation with respect to $o1$ can be described by two ZYX Euler angles ϕ_1 and θ_1 (rotation around Z and around Y respectively).

- $f2$ is placed on the moving sphere, in a similar way as $f1$ on the fixed sphere. The orientation of $f2$ with respect to $o2$ is described by ZYX Euler angles ϕ_2 and θ_2 (rotation around Z and around Y respectively).

The feature Jacobians are chosen such, that τ_1 and τ_2 correspond to $\dot{\phi}_1$ and $\dot{\theta}_1$ respectively, and τ_5 and τ_6 to $\dot{\phi}_2$ and $\dot{\theta}_2$ (this is, corresponding to the time rates of a minimal set of coordinates $\boldsymbol{\chi}$, as described in Section 5.2.2). The remaining two degrees of freedom reside between $f1$ and $f2$: translation along and rotation around the common normal to the spheres. The X-axis of $f1$ corresponds to this common normal. This leads to the following feature Jacobians:

$$\begin{aligned} {}_{o1}\mathbf{t}_{o1}^{f1} &= \boldsymbol{J_{FI}} \boldsymbol{\tau}_I, \\ &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -\sin(\phi_1) \\ 0 & \cos(\phi_1) \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}, \end{aligned} \tag{6.17}$$

$$\begin{aligned} {}_{f1}\mathbf{t}_{f1}^{f2} &= \boldsymbol{J_{FII}} \boldsymbol{\tau}_{II}, \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tau_3 \\ \tau_4 \end{bmatrix}, \end{aligned} \tag{6.18}$$

$$\begin{aligned} {}_{o2}\mathbf{t}_{o2}^{f2} &= \boldsymbol{J_{F\mathit{III}}}\boldsymbol{\tau_{\mathit{III}}}, \\ &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -\sin(\phi_2) \\ 0 & \cos(\phi_2) \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \tau_5 \\ \tau_6 \end{bmatrix}. \end{aligned} \tag{6.19}$$

The twists of $o1$ and $o2$ are given by:

$$ {}_{w}\mathbf{t}_{w}^{o1} = \mathbf{0}, \tag{6.20}$$

$$ {}_{w}\mathbf{t}_{w}^{o2} = \boldsymbol{J_R}\dot{\boldsymbol{q}}_{\boldsymbol{R}}. \tag{6.21}$$

Constraints are now defined to impose the motion of the spheres:

- Two position constraints realizing desired Euler angles $\phi_{1desired}$ and $\theta_{1desired}$ between $f1$ and $o1$:

$$\tau_1 = k_{fb}(\phi_{1desired} - \phi_{1actual}), \tag{6.22}$$

$$\tau_2 = k_{fb}(\theta_{1desired} - \theta_{1actual}), \tag{6.23}$$

- Two position constraints realizing desired Euler angles $\phi_{2desired}$ and $\theta_{2desired}$ between $f2$ and $o2$:

$$\tau_5 = k_{fb}(\phi_{2desired} - \phi_{2actual}), \tag{6.24}$$

$$\tau_6 = k_{fb}(\theta_{2desired} - \theta_{2actual}), \tag{6.25}$$

- One position constraints realizing the desired rotation of $f2$ around the X-axis of $f1$:

$$\tau_4 = k_{fb}(\psi_{desired} - \psi_{actual}), \tag{6.26}$$

- One force constraint, realizing the desired contact force $\boldsymbol{F}$:

$$\tau_3 = k_{fb}k_{st}^{-1}(\boldsymbol{F}_{desired} - \boldsymbol{F}_{actual}). \tag{6.27}$$

  In this, k_{st} is the contact stiffness. During the approach or retract phases, this constraint is replaced by a position constraint realizing a desired distance along the common normal (this is, the X-axis of $f1$):

$$\tau_3 = k_{fb}(x_{desired} - x_{actual}). \tag{6.28}$$

  Just before making contact, a transition is made from constraint (6.28) to (6.27) by changing the corresponding weights, as discussed in Section 5.4.

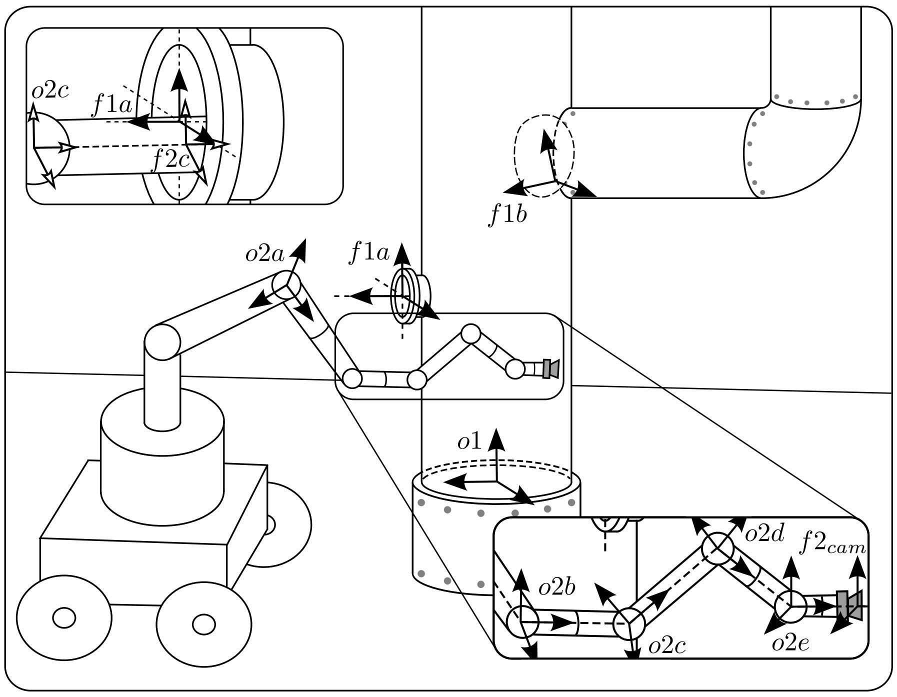


FIGURE 6.3: The object and feature frames for a maintenance task.

6.4 Inspection Task

This section deals with the second example task of Section 4.3.1 (Figure 4.4, page 51). In the task, a high-DOF maintenance robot enters a pipeline through an uncovered flange and points a camera to a seam to perform a visual check. The camera is attached to the last link of the robot.

The links of the robot should enter the flange one by one, starting with the last link of the robot. For every link-flange pair a specific set of constraints is needed when the link enters the flange. Furthermore, for the links inside the pipe other constraints are active to align the camera with the seam, and to avoid collisions between the links and the pipe. Hence this task consists of numerous states, during each of which different constraints are active.

In each state of the task, the pipe is one of the relevant objects. Hence, $o1$ is chosen at a reference position on the pipe (Figure 6.3). Two features of the pipe are relevant for the task: the flange, through which the robot enters the pipe, and the seam, with respect to which the camera should be aligned to perform a visual inspection. Hence a feature frame is attached to each of these features. The first feature frame, $f1a$, has its Y-axis aligned with the centerline of the flange. Its Z-axis is chosen vertically, and the origin is chosen in the middle of the flange. The second feature frame, $f1b$, is chosen at a reference position on the seam.

For the constraints regarding the positioning of a link with respect to the flange, the link is the second relevant object. Hence, an object frame is attached to each link which will enter the flange. More specifically, these frames are chosen at the base of the respective link, with the X-axis along the centerline of the link. For instance, for the robot in Figure 6.3, $o2e$ is the base frame of the last link of the robot, $o2d$ that of the one but last link, and so on[2]. For each of those links, the feature frame is chosen parallel to the respective object frame, but with its origin at the intersection point of the centerline of the link with the XZ-plane of $f1a$ (see Figure 6.3, inset top-left)[3]. The motion of these frames can be modeled by the following feature Jacobians:

- There is no motion of $f1a$ with respect to $o1$:

$$\begin{aligned} \mathbf{t}_{o1}^{f1a} &= \boldsymbol{J}_{\boldsymbol{FI}}^{a}\boldsymbol{\tau_{I}}^{a}, \\ &= \mathbf{0}. \end{aligned} \tag{6.29}$$

- The feature frame of a link i, $f2i$, has 5 DOF with respect to $f1a$: motion of the origin of $f2i$ in the XZ-plane of $f1a$ (2 DOF), and rotation of $f2i$ with respect to $f1a$ (3 DOF). The following feature Jacobian reflects these DOF:

$$\begin{aligned} {}_{f1a}^{f2i}\mathbf{t}_{f1a}^{f2i} &= \boldsymbol{J}_{\boldsymbol{FII}}^{a}\boldsymbol{\tau_{II}}^{a}, \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_1^a \\ \tau_2^a \\ \tau_3^a \\ \tau_4^a \\ \tau_5^a \end{bmatrix}. \end{aligned} \tag{6.30}$$

[2]The naming scheme is somewhat different than usual, as there are so many features. Regarding the robot entering the pipe, first $o1$, $f1a$, $o2e$ and $f2e$ are the relevant frames, as the last link of the robot enters the flange. In the second state, when the one but last link enters the flange, the relevant frames are $o1$, $f1a$, $o2d$ and $f2d$. And so on.

[3]The inset depicts a link which sits through the flange. However, the intersection point is always defined when the centerline is not parallel to XZ-plane of $f1a$, also when the link is not through the flange.

- $f2i$ translates along the X-axis of the object frame of link i, $o2i$ (1 DOF):

$$\begin{aligned} {}_{o2i}\mathbf{t}_{o2i}^{f2i} &= \boldsymbol{J}_{\boldsymbol{F}\boldsymbol{I\!I\!I}}^{a} \boldsymbol{\tau}_{\boldsymbol{I\!I\!I}}{}^{a}, \\ &= \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \tau_6^a \end{bmatrix}. \end{aligned} \tag{6.31}$$

For the constraints regarding the positioning of the camera with respect to the seam, the last link of the robot is the relevant object, with object frame $o2e$, and the relevant feature is the camera, with feature frame $f2cam$. $f2cam$ is fixed with respect to $o2e$, and $f1b$ is fixed with respect to $o1$. Hence all 6 DOF are present between $f2cam$ and $f1b$:

$$\begin{aligned} \mathbf{t}_{o1}^{f1b} &= \boldsymbol{J}_{\boldsymbol{F}\boldsymbol{I}}^{b} \boldsymbol{\tau}_{\boldsymbol{I}}{}^{b}, \\ &= \mathbf{0}, \end{aligned} \tag{6.32}$$

$$\begin{aligned} {}_{f1b}^{f2b}\mathbf{t}_{f1b}^{f2cam} &= \boldsymbol{J}_{\boldsymbol{F}\boldsymbol{I\!I}}^{b} \boldsymbol{\tau}_{\boldsymbol{I\!I}}{}^{b}, \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_1^b \\ \tau_2^b \\ \tau_3^b \\ \tau_4^b \\ \tau_5^b \\ \tau_6^b \end{bmatrix}, \end{aligned} \tag{6.33}$$

$$\begin{aligned} \mathbf{t}_{o2e}^{f2cam} &= \boldsymbol{J}_{\boldsymbol{F}\boldsymbol{I\!I\!I}}^{b} \boldsymbol{\tau}_{\boldsymbol{I\!I\!I}}{}^{b}, \\ &= \mathbf{0}. \end{aligned} \tag{6.34}$$

For each link i, a robot Jacobian $\boldsymbol{J}_{\boldsymbol{R}i}$ relates the robot joint velocities to the twist of $o2i$:

$$ {}_{w}\mathbf{t}_{w}^{o2i} = \boldsymbol{J}_{\boldsymbol{R}i} \dot{\boldsymbol{q}}_{\boldsymbol{R}}. \tag{6.35}$$

The Jacobian till the last link is $\boldsymbol{J}_{\boldsymbol{R}e}$. Each of the Jacobians to an intermediate link i is easily found from $\boldsymbol{J}_{\boldsymbol{R}e}$ by copying its first i columns, and setting the other columns to $\boldsymbol{0}_{6\times 1}$.

For each state constraints are now defined to realize the task. It is assumed that the task starts with the robot's last link in the neighborhood of the flange, and with the centerline of the last link not parallel to the XZ-plane of $f1a$. For instance, the robot can reach this position by means of jointspace motion. In the first phase, the last link of the robot enters the flange. In each consecutive phase, a further link enters the flange, and the links inside the pipe move to reach the desired pose of the camera with respect to the seam.

Phase 1

The first phase only concerns the motion of the last link of the robot with respect to the flange, so the relevant object and feature frames are $o1$, $f1a$, $f2e$ and $o2e$. The following constraints realize the desired motion of the robot:

- To keep the link centered with respect to the flange, the origin of $f2e$ is regulated to a central position in the XZ-plane of $f1a$ (for instance, so that it coincides with the origin of $f1a$):

$$\begin{aligned} \tau_1^a &= k_{fb}(x_{desired} - x_{actual}), & (6.36) \\ \tau_2^a &= k_{fb}(z_{desired} - z_{actual}). & (6.37) \end{aligned}$$

  In this, $x_{desired}$ and $z_{desired}$ are the desired x and z coordinates of the origin of $f2e$ with respect to $f1a$, and x_{actual} and z_{actual} their actual values.

- The orientation of $f2e$ is regulated so, that the centerlines of the link and the flange coincide. For this, control constraints on the ZYX Euler angles ϕ and θ (rotation around Z and Y respectively) of $f2e$ with respect to $f1a$ are needed[4]. As τ_3^a, τ_4^a and τ_5^a are defined in terms of rotational velocities, integrating factors are present in these constraints:

$$\begin{bmatrix} \frac{\cos\phi\sin\theta}{\cos\theta} & \frac{\sin\phi\sin\theta}{\cos\theta} & 1 \\ -\sin\phi & \cos\phi & 0 \end{bmatrix} \begin{bmatrix} \tau_3^a \\ \tau_4^a \\ \tau_5^a \end{bmatrix} = \begin{bmatrix} k_{fb}(\phi_{desired} - \phi_{actual}) \\ k_{fb}(\theta_{desired} - \theta_{actual}) \end{bmatrix}. \qquad (6.38)$$

- A further constraint is placed on the X-motion of $f2e$ with respect to $o2e$, so the link enters the pipe:

$$\tau_6^a = k_{fb}(x'_{desired} - x'_{actual}). \qquad (6.39)$$

  In this, $x'_{desired}$ is the desired x coordinates of the origin of $f2e$ with respect to $o2e$, and x'_{actual} its actual value.

A trajectory generator provides the desired values for x, z, ϕ, θ and x', according to a smooth trajectory from the initial value to the desired final value (for instance, according to a trapezoidal velocity profile).

The five control constraints are sufficient to realize the insertion motion of the last link into the pipe. However, the complete robot motion is still underconstrained. For collision avoidance between the links and the pipe, extra

[4]Note that $f1a$ and $f2e$ were chosen such (this is, with Y and X-axis along the centerline of the flange and the link, respectively), that the representation of their relative orientation by ZYX Euler angles is well conditioned when the link sits through the flange.

jointspace constraints are added in the nullspace of above constraints, which keep the robot close to a neutral stance. For each link i, such a constraint is given by:

$$\dot{q}_i = k_{fb}(q_{i\,desired} - q_{i\,actual}). \tag{6.40}$$

For a large diameter of the pipe relative to that of the links, this approach is sufficient to ensure collision avoidance. An alternative, more reliable approach is to define constraints on the distance between the links and the pipe. However, this approach requires the definition of extra object and feature frames and is hence somewhat more involved.

Phase 2

The first phase is concluded when the last link of the robot has entered the pipe, this is, when $x'_{actual} = 0$. In the second phase, most constraints of the first phase are copied to the next link, this is, to the relative motion of $o1$, $f1a$, $f2d$ and $o2d$:

- The X and Z-positions of $f2d$ with respect to $f1a$ are regulated, and
- The ϕ and θ Euler angles of $f2d$ with respect to $f1a$ are regulated.

However, the other constraint, on the X-position of $f2d$ with respect to $o2d$, is not copied. Instead of this constraint, a desired pose of $f2cam$ with respect to $f1b$ is specified, for instance by constraints on the relative position (three constraints) and on the Euler angles of $f2cam$ with respect to $f1b$ (another three constraints). As there is only one link inside the robot, these *secondary* constraints cannot be fully realized without breaking those realizing the relative pose of the link through the flange (the *primary* constraints). Hence, to prevent this interference between the constraints, the secondary constraints are specified in the nullspace of the primary constraints. Furthermore, as the part of the robot inside the pipe has less than six DOF, the six secondary constraints will be conflicting. Hence, weights in constraint space are needed. As the camera is still far away from the flange, the position constraints are more important than the constraints on the orientation. Hence, a good choice is to put a higher weight on the position constraints than on the orientation constraints.

While the secondary constraints overconstrain the part of the robot inside the pipe, the motion of that part of the robot which is outside the pipe is not necessarily fully constrained. Hence, also in this phase extra jointspace constraints (6.40) are specified, in the nullspace of both the primary and the secondary constraints.

Consecutive phases

In the consecutive phases, similar constraints as in phase 2 are active, but each time on the next link. For instance, in phase 3 the constraints are related to $o1$, $f1a$, $f2c$ and $o2c$, in phase 4 to $o1$, $f1a$, $f2b$ and $o2b$ and so on. The constraints on the relative pose of $f2cam$ and $o2d$ stay active. At the end, the robot has entered the pipe as far as it should, and enough DOF are available inside the pipe to align the camera and the seam as desired. In this case, the weights on the constraints regarding the camera and the seam become irrelevant, as the constraints are no longer conflicting.

6.5 Incompatible Seam Following (Multi-point contact)

A common example of a task which is not easily expressible using the TFF is *incompatible seam following* or *multi-point contact* (Figure 6.4). This section shows that this task can easily be specified using the formalism presented in this thesis. Seam following is the generalization of 2D contour following to three dimensions and involves controlled motion of a tool along a seam, which is formed by two surfaces in the environment. In general, the tool's geometry is incompatible with the seam, which leads to two point contacts between the tool and the environment. In this experiment, the surfaces are planar. Hence, two vertex-face contacts occur, and two contact forces are regulated. The other four DOF of the tool with respect to the seam are constrained by motion specifications: a desired relative orientation of the tool and the seam (2 DOF), and a desired trajectory of one of the contact points in the contact plane (2 DOF).

The two vertex-face contacts suggest two features: the contact point of the first contact (feature a), and the contact point of the second contact (feature b). The related objects for each feature are the tool and the corresponding surface. The frames are chosen as follows (Figure 6.4):

- $o1a$ and $o1b$ are reference frames in the contacting plane of feature a and feature b respectively. They have their Z-axis along the normal of the corresponding plane.
- $o2$ is a reference frame on the mounting plate of the robot.
- $f1a$ and $f1b$ have their origin at the contacting vertex of the tool, and the same orientation as $o1a$ and $o1b$ respectively.
- $f2a$ and $f2b$ also have their origin at the contacting vertices, but the same orientation as $o2$.

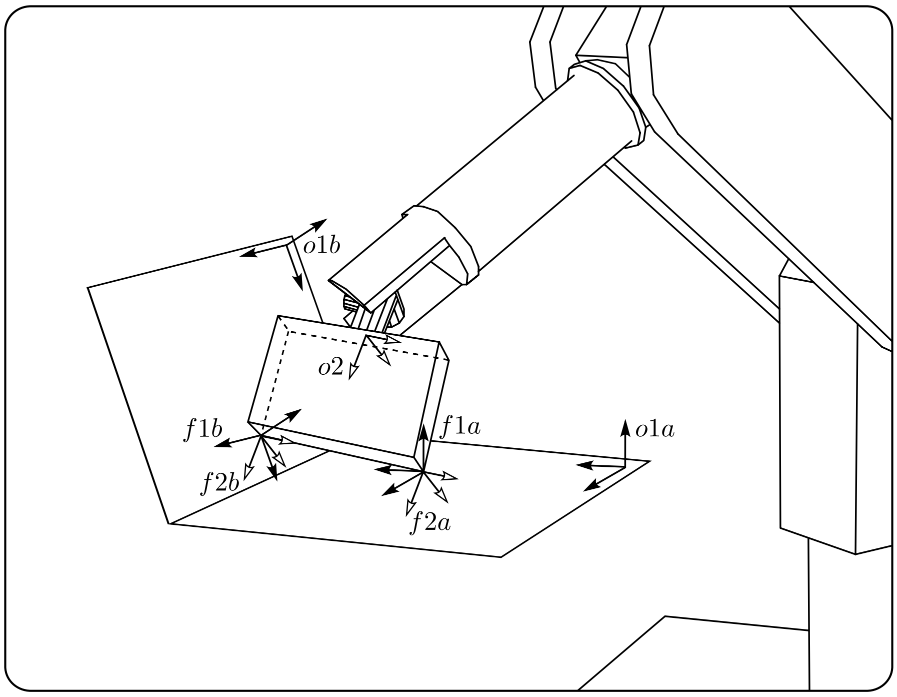


FIGURE 6.4: The object and feature frames for an incompatible seam following task.

The following feature Jacobians represent the submotions of the object and feature frames:

- $f1a$ and $f1b$ translate with respect to $o1a$ and $o1b$ respectively:

$$
\begin{aligned}
{}_{o1a}\mathbf{t}_{o1a}^{f1a} &= \boldsymbol{J}_{\boldsymbol{FI}}^{a}\boldsymbol{\tau}_{I}{}^{a}, \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tau_1^a \\ \tau_2^a \\ \tau_3^a \end{bmatrix},
\end{aligned}
\tag{6.41}
$$

$$
\begin{aligned}
{}_{o1b}\mathbf{t}_{o1b}^{f1b} &= \boldsymbol{J}_{\boldsymbol{F}I}^{b}\boldsymbol{\tau}_{I}{}^{b}, \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tau_1^b \\ \tau_2^b \\ \tau_3^b \end{bmatrix}, \qquad (6.42)
\end{aligned}
$$

- $f1a$ and $f1b$ rotate with respect to $f2a$ and $f2b$ respectively:

$$
\begin{aligned}
{}_{f1a}\mathbf{t}_{f1a}^{f2a} &= \boldsymbol{J}_{\boldsymbol{F}II}^{a}\boldsymbol{\tau}_{II}{}^{a}, \\
&= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_4^a \\ \tau_5^a \\ \tau_6^a \end{bmatrix}, \qquad (6.43)
\end{aligned}
$$

$$
\begin{aligned}
{}_{f1b}\mathbf{t}_{f1b}^{f2b} &= \boldsymbol{J}_{\boldsymbol{F}II}^{b}\boldsymbol{\tau}_{II}{}^{b}, \\
&= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_4^b \\ \tau_5^b \\ \tau_6^b \end{bmatrix}, \qquad (6.44)
\end{aligned}
$$

- There is no motion between $f2a$, $f2b$ and $o2$:

$$
\begin{aligned}
{}_{o2}\mathbf{t}_{o2}^{f2a} &= \boldsymbol{J}_{\boldsymbol{F}III}^{a}\boldsymbol{\tau}_{III}{}^{a}, \\
&= \mathbf{0}, \qquad (6.45) \\
{}_{o2}\mathbf{t}_{o2}^{f2b} &= \boldsymbol{J}_{\boldsymbol{F}III}^{b}\boldsymbol{\tau}_{III}{}^{b}, \\
&= \mathbf{0}. \qquad (6.46)
\end{aligned}
$$

The twists of $o1$ and $o2$ are given by:

$$
\begin{aligned}
{}_{w}\mathbf{t}_{w}^{o1} &= \mathbf{0}, \qquad (6.47) \\
{}_{w}\mathbf{t}_{w}^{o2} &= \boldsymbol{J}_{\boldsymbol{R}}\dot{\boldsymbol{q}}_{\boldsymbol{R}}. \qquad (6.48)
\end{aligned}
$$

As previously mentioned, two force constraints are needed to regulate the contact force in each contact. Furthermore, two constraints impose the motion of the contact point in the contacting plane, and two constraints regulate the orientation of the tool. This leads to the following constraints:

- Two constraints regarding the contact forces $\boldsymbol{F^i}$:

$$\begin{aligned}\tau_3^a &= k_{fb}{k_{st}^a}^{-1}(\boldsymbol{F^a}_{desired} - \boldsymbol{F^a}_{actual}), & (6.49)\\ \tau_3^b &= k_{fb}{k_{st}^b}^{-1}(\boldsymbol{F^b}_{desired} - \boldsymbol{F^b}_{actual}), & (6.50)\end{aligned}$$

  in which k_{st}^a and k_{st}^b are the respective contact stiffnesses,

- Two constraints regulating the position of the contact point:

$$\begin{aligned}\tau_1^a &= k_{fb}(x_{desired}^a - x_{actual}^a), & (6.51)\\ \tau_2^a &= k_{fb}(y_{desired}^a - y_{actual}^a), & (6.52)\end{aligned}$$

  in which x_{actual}^a and y_{actual}^a are the x and y coordinate of $f1a$ with respect to $o1a$, and $x_{desired}^a$ and $y_{desired}^a$ their desired values,

- Two constraints on the orientation of the tool, for instance by regulating the two Euler angles ϕ and θ (around Z and Y respectively) between $f1a$ and $f2a$:

$$\begin{aligned}&\begin{bmatrix} \dfrac{\cos\phi\sin\theta}{\cos\theta} & \dfrac{\sin\phi\sin\theta}{\cos\theta} & 1 \\ -\sin\phi & \cos\phi & 0 \end{bmatrix}\begin{bmatrix}\tau_4^a\\ \tau_5^a\\ \tau_6^a\end{bmatrix}\\ &\qquad = \begin{bmatrix} k_{fb}(\phi_{desired} - \phi_{actual}) \\ k_{fb}(\theta_{desired} - \theta_{actual})\end{bmatrix}. \qquad (6.53)\end{aligned}$$

$\boldsymbol{J^a_{F\mathit{II}}}$, chosen in (6.43), defines τ_4^a, τ_5^a and τ_6^a as rotational velocities. As these do not integrate into a set of Euler angles without integrating factors, these integrating factors are present in the constraint (6.53). As the constraint concerns the Euler angles ϕ and θ the integrating factors correspond to the first two rows of $\boldsymbol{E}^{-1}$. An alternative is to define $\boldsymbol{J^a_{F\mathit{II}}}$ such, that τ_4^a, τ_5^a and τ_6^a correspond directly to Euler angle rates:

$$\begin{aligned}{}_{f1a}\mathbf{t}_{f1a}^{f2a} &= \boldsymbol{J^a_{F\mathit{II}}}\boldsymbol{\tau_{\mathit{II}}}^a,\\ &= \begin{bmatrix} 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & -\sin\phi & \cos\theta\cos\phi\\ 0 & \cos\phi & \cos\theta\sin\phi\\ 1 & 0 & -\sin\phi\end{bmatrix}\begin{bmatrix}\tau_3^a\\ \tau_4^a\\ \tau_5^a\end{bmatrix}. \qquad (6.54)\end{aligned}$$

In that case, the constraint (6.53) simplifies to:

$$\begin{bmatrix}\tau_4^a\\ \tau_5^a\end{bmatrix} = \begin{bmatrix} k_{fb}(\phi_{desired} - \phi_{actual})\\ k_{fb}(\theta_{desired} - \theta_{actual})\end{bmatrix}. \qquad (6.55)$$

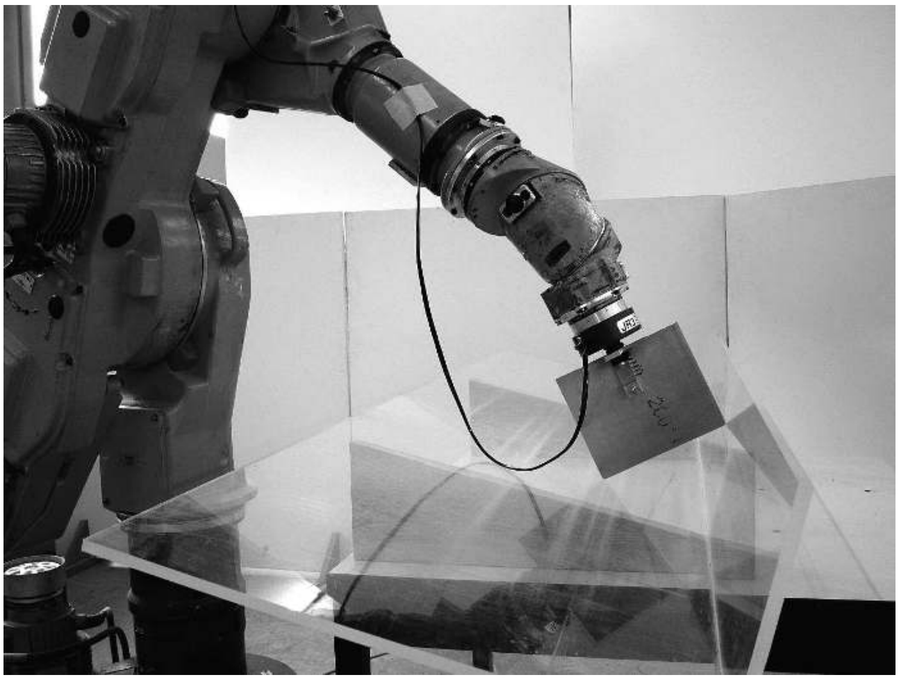


Figure 6.5: The experimental setup of the incompatible seam following experiment.

The incompatible seam following task was performed on a Kuka 361 industrial robot, equipped with a wrist-mounted JR3 force/torque sensor (Figure 6.5). In the experiment, the planes of the seam form an angle of 90 degrees. First, contact is established, with desired contact forces of 10N in each contact. Then, a sinusoidal motion of the tool along the seam is commanded (unit: [m]):

$$\begin{aligned} x_{des} &= 0.08 + 0.03\sin(\frac{2\pi}{7.5}t), \\ y_{des} &= 0.15\cos(\frac{2\pi}{15}t + \pi). \end{aligned}$$

During the experiment, the tool is kept perpendicular to the seam. Figures 6.6 and 6.7 show the measured contact forces. The plots clearly show the disturbance of friction on the force control, when the sinusoidal motion is started, around 50s.

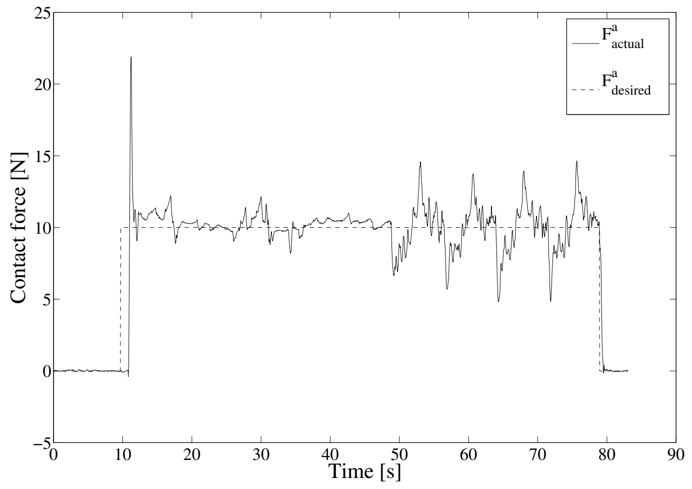


FIGURE 6.6: The contact force in contact a. From $50s$ on, the tool moves sinusoidally along the seam. The disturbance of friction on the contact force can be seen.

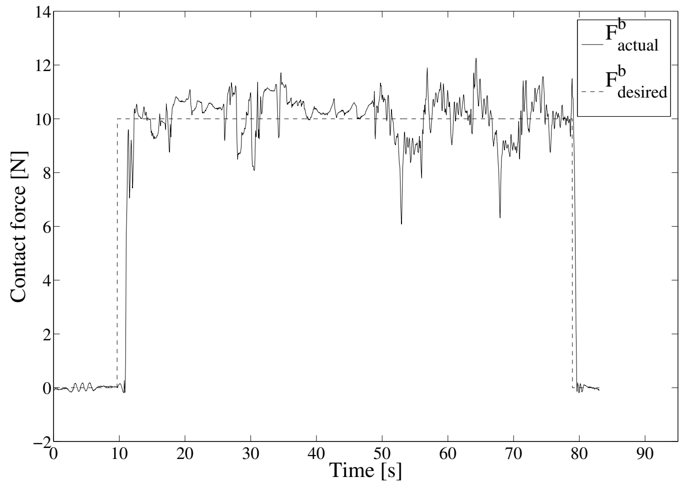


FIGURE 6.7: The contact force in contact b. From $50s$ on, the tool moves sinusoidally along the seam. The disturbance of friction on the contact force is less prominent than in contact a, but can still be seen.

6.6 Laser Tracing

This section shows the application of the methodology to a geometrically complex task involving an underconstrained specification as well as estimation of uncertain geometric parameters. The goal is to simultaneously trace a path on a plane as well as on a cylindrical barrel using two lasers which are rigidly attached to the robot end effector (Figure 6.8). The lasers also measure the distance to the surface. The exact position and orientation of the plane are initially unknown. The barrel is standing in a vertical position on a floor so its vertical position is known. However, its exact position on the floor is unknown.

The use of two lasers beams suggests the use of two feature relationships, one for the laser-plane combination, feature a, and one for the laser-barrel combination, feature b. Figure 6.8 shows the object and feature frames. For the laser-plane feature:

- frame $o1a$ is fixed to the plane, with its Z-axis perpendicular to the plane,
- frame $o2a$ is fixed to the first laser on the robot end effector, with its Z-axis along the laser beam,
- frame $f1a$ has the same orientation as $o1a$, but is located at the laser spot (this is, the intersection of the laser beam and the plane),
- frame $f2a$ is also located at the laser spot, but has the same orientation as $o2a$,

and for the laser-barrel feature:

- frame $o1b$ is fixed to the barrel, with its X-axis along the axis of the barrel,
- frame $o2b$ is fixed to the second laser on the robot end effector, with its Z-axis along the laser beam,
- frame $f1b$ has its origin at the laser spot on the barrel, and is oriented with its Z-axis perpendicular to the barrel surface and its X-axis parallel to the barrel axis,
- frame $f2b$ has its origin at the same position as that of $f1b$, and has the same orientation as $o2b$.

The following feature Jacobians represent the submotions of the object and feature frames for feature a:

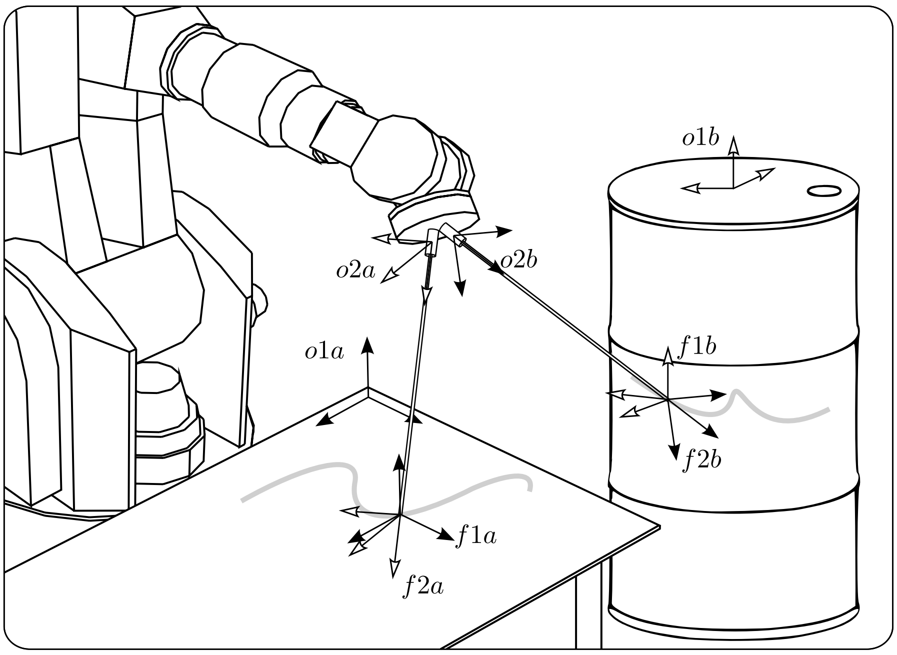


FIGURE 6.8: The object and feature frames for a laser tracing task. The goal of the task is to trace simultaneously a path on a plane as well as on a cylindrical barrel using two lasers which are rigidly attached to the robot end effector. The lasers also measure the distance to the surface. The position and orientation of the plane and the position of the barrel are uncertain at the beginning of the task.

- $f1a$ can translate on the plane, that is, in the XY-plane of $o1a$:

$$\begin{aligned} {}_{o1a}\mathbf{t}^{f1a}_{o1a} &= \boldsymbol{J}^{a}_{\boldsymbol{FI}}\boldsymbol{\tau}_{I}{}^{a}, \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tau^{a}_{1} \\ \tau^{a}_{2} \end{bmatrix}, \end{aligned} \tag{6.56}$$

- $f2a$ rotates with respect to $f1a$:

$$\begin{aligned} {}_{f1a}\mathbf{t}_{f1a}^{f2a} &= \boldsymbol{J}_{\boldsymbol{F}\mathit{II}}^{a}\boldsymbol{\tau}_{\mathit{II}}{}^{a}, \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_3^a \\ \tau_4^a \\ \tau_5^a \end{bmatrix}, \end{aligned} \tag{6.57}$$

- $f2a$ translates along the laser beam, this is, along the Z-axis of $o2a$:

$$\begin{aligned} {}_{o2a}\mathbf{t}_{o2a}^{f2a} &= \boldsymbol{J}_{\boldsymbol{F}\mathit{III}}^{a}\boldsymbol{\tau}_{\mathit{III}}{}^{a}, \\ &= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \tau_6^a \end{bmatrix}, \end{aligned} \tag{6.58}$$

and for feature b:

- $f1b$ can translate and rotate along the X-axis of $o1b$:

$$\begin{aligned} {}_{o1b}\mathbf{t}_{o1b}^{f1b} &= \boldsymbol{J}_{\boldsymbol{F}\mathit{I}}^{b}\boldsymbol{\tau}_{\mathit{I}}{}^{b}, \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tau_1^b \\ \tau_2^b \end{bmatrix}, \end{aligned} \tag{6.59}$$

- $f2b$ rotates with respect to $f1b$:

$$\begin{aligned} {}_{f1b}\mathbf{t}_{f1b}^{f2b} &= \boldsymbol{J}_{\boldsymbol{F}\mathit{II}}^{b}\boldsymbol{\tau}_{\mathit{II}}{}^{b}, \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_3^b \\ \tau_4^b \\ \tau_5^b \end{bmatrix}, \end{aligned} \tag{6.60}$$

- $f2b$ translates along the laser beam, this is, along the Z-axis of $o2b$:

$$\begin{aligned} {}_{o2b}\mathbf{t}_{o2b}^{f2b} &= \boldsymbol{J}_{\boldsymbol{F}\mathit{III}}^{b}\boldsymbol{\tau}_{\mathit{III}}{}^{b}, \\ &= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \tau_6^b \end{bmatrix}. \end{aligned} \tag{6.61}$$

The twists of $o1i$ and $o2i$, with $i = a$ or b, are given by:

$${}_{w}\mathbf{t}_{w}^{o1i} = \mathbf{0}, \tag{6.62}$$

$${}_{w}\mathbf{t}_{w}^{o2i} = \boldsymbol{J_R}\dot{\boldsymbol{q}}_{\boldsymbol{R}}. \tag{6.63}$$

The following constraints ensure that the paths are traced on the plane and the barrel:

- To trace a path on the plane, constraints are needed on the X and Y-motion of $f1a$ with respect to $o1a$:

$$\tau_1^a = k_{fb}(x_{desired} - x_{actual}), \tag{6.64}$$

$$\tau_2^a = k_{fb}(y_{desired} - y_{actual}), \tag{6.65}$$

  with x_{actual} and y_{actual} the X and Y-position of $f1a$ with respect to $o1a$, and $x_{desired}$ and $y_{desired}$ the desired values for these positions.

- As τ_1^b and τ_2^b correspond to cylinder coordinate rates, the following constraints realize a desired path on the barrel:

$$\tau_1^b = k_{fb}(x'_{desired} - x'_{actual}), \tag{6.66}$$

$$\tau_2^b = k_{fb}(\psi_{desired} - \psi_{actual}), \tag{6.67}$$

  with x'_{actual} the X-position of $f1b$ with respect to $o1b$, ψ_{actual} its rotation around the X-axis of $o1b$, and $x'_{desired}$ and $\psi_{desired}$ their desired values.

These constraints realize the desired motion of the laser spots on the plane and the barrel. The complete robot motion however is still underconstrained. A possible solution is to define weights in jointspace. The weighted pseudo-inverse method then yields those values for $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ with minimal weighted norm. Another possibility is to specify extra desired joint values, by specifying constraints

$$\dot{q}_i = k_{fb}(q_{i\,desired} - q_{i\,actual}) \tag{6.68}$$

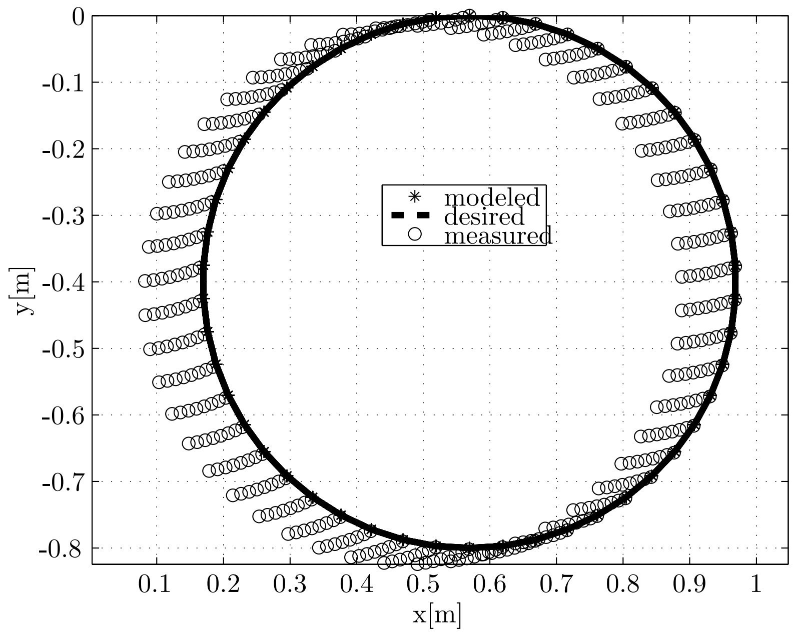


Figure 6.9: Motion of the laser spot on the plane, without model correction (by courtesy of Wilm Decré). Without model correction, the iterative application of prediction results in accumulation of integration errors due to discretization and due to errors in the geometric model. Therefore the estimates of the feature frames drift away from their actual positions. While the estimated position of the laser spot corresponds well to its desired position, the actual position does not.

in the nullspace of the constraints (6.64)–(6.67).

Figures 6.9 and 6.10 show some simulation results of the motion of the laser spot on the plane (feature a), in the case of a known plane and barrel position. The values for $x_{desired}$ and $y_{desired}$ are chosen such that the laser spot performs a circular trajectory in the plane, with a radius of $0,4m$. Nullspace constraints (6.68) are applied, so the robots stays close to its nominal position. These simulation results illustrate the model prediction and correction steps (Section 5.2). In the results of Figure 6.9, no model correction is performed: the object and feature frame poses are only determined by the model prediction. The iterative application of the prediction results in accumulation of integration errors due to discretization and due to errors in the geometric model. Therefore the estimates of the feature frames drift

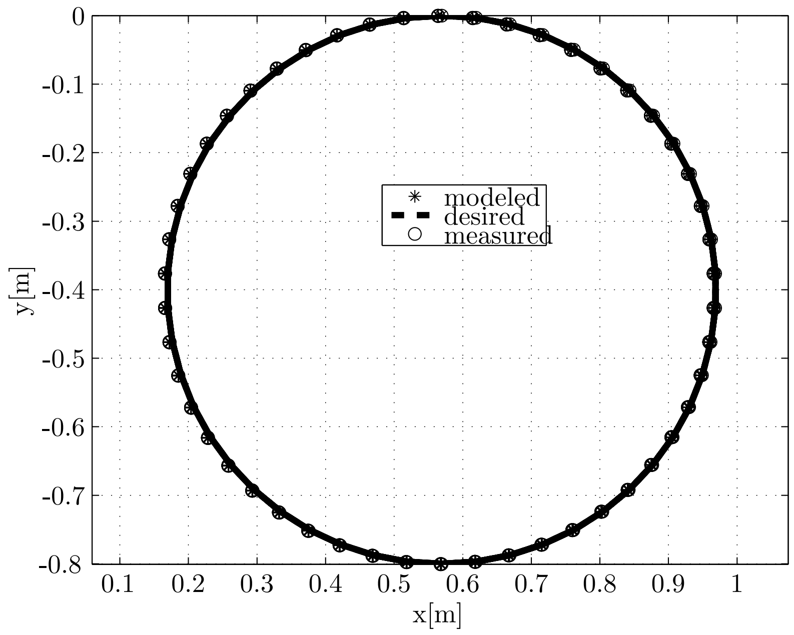


FIGURE 6.10: Motion of the laser spot on the plane, with model correction (by courtesy of Wilm Decré). If the correction step is added, based on the additional information contained in the pose closure equations, the drift of the estimates is prevented. The estimated laser spot position corresponds to its actual and desired position.

away from their actual positions: the origin of $f1a$ no longer coincides with the real laser spot on the plane. If however the correction step based on the additional information contained in the pose closure equations is added, the drift of the frames is prevented, as shown in Figure 6.10.

Laser tracing with distance control

As the laser sensors measure the distance to the plane and the cylinder, the current task can be expanded with extra constraints concerning these distances. For instance, it is important to keep the lasers as good as possible at the nominal distance to the plane and the cylinder, as the lasers measurement ranges are limited. Another example is that of spray painting. Spray painting has the same topology as laser tracing (with one laser). However, for spray painting it is important that the spray gun is maintained at a fixed distance to the painted object. To this extent, two constraints are added:

$$\tau_6^a = k_{fb}(z_{desired} - z_{actual}), \tag{6.69}$$

$$\tau_6^b = k_{fb}(z'_{desired} - z'_{actual}), \tag{6.70}$$

with $z_{desired}$ and $z'_{desired}$ the desired distances of the lasers to the plane and the cylinder respectively, and z_{actual} and z'_{actual} their actual values. Adding two extra constraints on the laser distances however makes the task overconstrained, as the constraints are conflicting. Hence, to ensure that the laser spots do not deviate from their desired trajectories, the distance constraints are applied in the nullspace of the primary constraints (6.64)–(6.67).

Laser tracing with estimation

The distance measurements from the laser sensors provide extra information about the pose of the objects with respect to the world. Hence, using these measurements it is possible to estimate these poses.

According to the procedure described in Section 5.3, extra coordinates $\boldsymbol{\chi_U}$ are introduced which describe the unknown position of the plane and the barrel. For the plane, these coordinates are chosen as:

$$\boldsymbol{\chi_U}^a = \begin{bmatrix} h^a \\ \theta^a \\ \psi^a \end{bmatrix}, \tag{6.71}$$

with h^a the z-coordinate in w of the intersection point of the plane and the Z-axis of w, and θ^a and ψ^a the Y and X-Euler angles respectively, which determine the orientation of the plane with respect to the world. The unknown position of the cylinder is modeled by:

$$\boldsymbol{\chi_U}^b = \begin{bmatrix} x^b \\ y^b \end{bmatrix}, \tag{6.72}$$

with x^b and y^b the X and Y-position of the cylinder with respect to w. Note that other parameters, such as the Z-Euler angle of the plane or the rotation of the cylinder around its axis are not observable and are not chosen part of the uncertainty coordinates.

The motion of the object frames can now be modeled as:

$${}_w\mathbf{t}_w^{o1a} = \boldsymbol{J}_{\boldsymbol{U}}^a \dot{\boldsymbol{\chi}}_{\boldsymbol{U}}^a \tag{6.73}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & \cos(\theta^a) \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{h}^a \\ \dot{\theta}^a \\ \dot{\psi}^a \end{bmatrix}, \tag{6.74}$$

and

$$
\begin{aligned}
{}_{w}\mathbf{t}_{w}^{o1b} &= \boldsymbol{J}_{\boldsymbol{U}}^{b} \dot{\boldsymbol{\chi}}_{\boldsymbol{U}}{}^{b} && (6.75)\\
&= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x^b \\ y^b \end{bmatrix}. && (6.76)
\end{aligned}
$$

Next, the prediction equation is determined. As, in this case, the object poses are constant, the prediction equation is given by:

$$
\widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}k+1} = \widehat{\boldsymbol{\chi}}_{\boldsymbol{U}k}, \tag{6.77}
$$

with $\boldsymbol{\chi}_{\boldsymbol{U}}$ the total coordinate vector:

$$
\boldsymbol{\chi}_{\boldsymbol{U}} = \begin{bmatrix} \boldsymbol{\chi}_{\boldsymbol{U}}{}^{a} \\ \boldsymbol{\chi}_{\boldsymbol{U}}{}^{b} \end{bmatrix}. \tag{6.78}
$$

The measurements z^a and z^b correspond to the Z-coordinates of $f2a$ with respect to $o2a$ and of $f2b$ with respect to $o2b$:

$$
\begin{aligned}
z^a &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{d}_{o2a}^{f2a}, && (6.79)\\
z^b &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{d}_{o2b}^{f2b}. && (6.80)
\end{aligned}
$$

Hence:

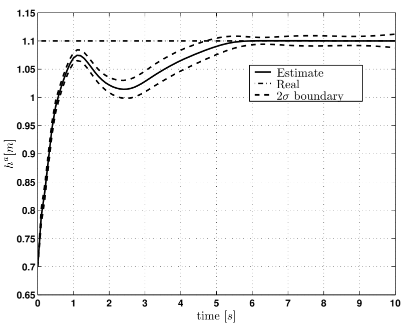
$$
\dot{\boldsymbol{z}} = \begin{bmatrix} \dot{z}^a \\ \dot{z}^b \end{bmatrix} = \begin{bmatrix} \tau_6^a \\ \tau_6^b \end{bmatrix}, \tag{6.81}
$$

or, with $\boldsymbol{H}_{\boldsymbol{F}}$ and $\boldsymbol{H}_{\boldsymbol{U}}$ according to (5.35):

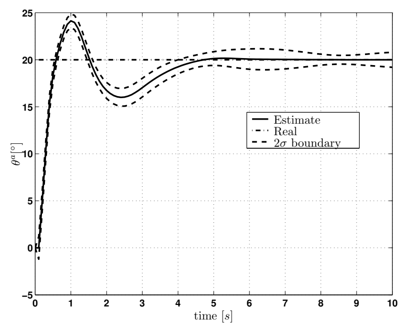
$$
\begin{aligned}
\boldsymbol{H}_{\boldsymbol{F}} &= \boldsymbol{0}_{2\times 12}, && (6.82)\\
\boldsymbol{H}_{\boldsymbol{U}} &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. && (6.83)
\end{aligned}
$$

According to (5.38), the complete Kalman equation $\boldsymbol{z} = \boldsymbol{H}\boldsymbol{x}$ is then obtained, and the parameters estimated according to the Kalman Filter procedures.

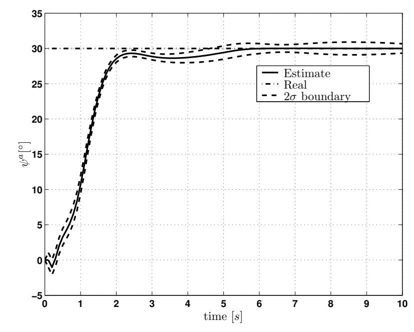
Simulations are shown in Figure 6.11 (plane) and Figure 6.12 (barrel). The initial estimation errors for the plane are $0.40m$ for h^a, 20° for Euler angle θ^a and 30° for Euler angle ψ^a. The initial estimation errors for the barrel are $0.40m$ in the X-direction and $0.10m$ in the Y-direction. An extended Kalman filter is used for the estimation. As soon as the locations of the plane and the barrel are estimated correctly, after tracing approximately one circle ($8s$), the circles traced by the laser beams equal the desired ones.



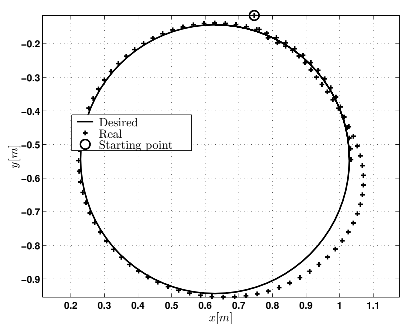
(a) Height of the plane, h^a.



(b) Orientation of the plane, θ^a.



(c) Orientation of the plane, ψ^a.



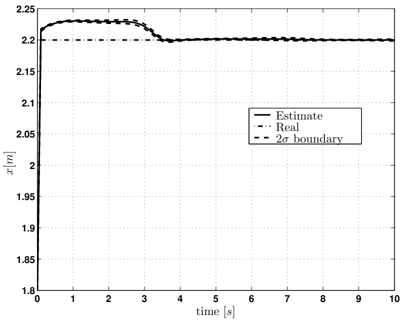
(d) Laser spot on plane.

Figure 6.11: Laser tracing on plane (simulation): estimation of plane height (h^a) and plane orientation (θ^a and ψ^a). (by courtesy of Tinne De Laet and Wilm Decré)

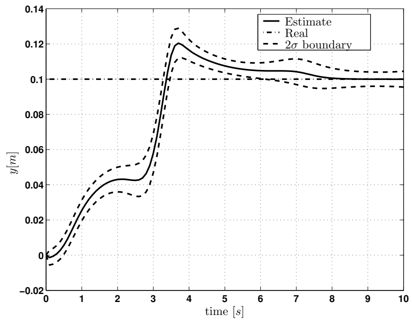
Figure 6.13 shows results of an experiment in which the position of a barrel position was estimated from distance measurements of a Baumer laser distance sensor (OADM 2016480/S14F). The figure shows the motion of the laser with respect to the barrel, the distance to the surface measured by the laser, and the estimation results. The position of the barrel is estimated with the desired accuracy after approximately $7s$.

6.7 Human-Robot Shared Control

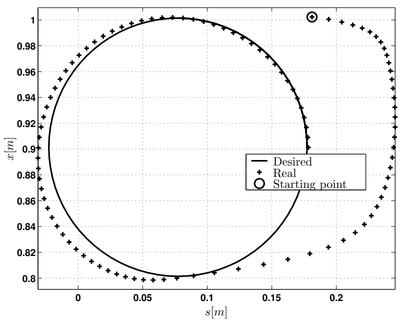
This section shows the application of the methodology to an overconstrained task involving human-robot co-manipulation (Figure 6.14). A robot assists an operator to carry a heavy machine part, and fine position it on a support



(a) x-position of the barrel, x^b.



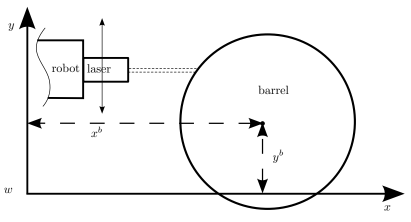
(b) y-position of the barrel, y^b.



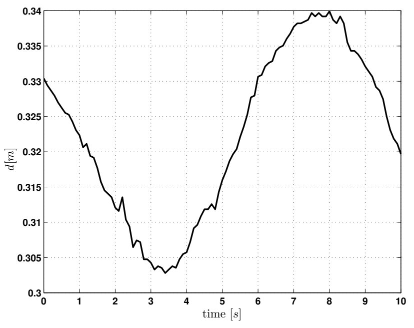
(c) Laser spot on barrel surface, with $s = R\alpha^b$ the arc length and x the distance along the cylinder axis.

FIGURE 6.12: Laser tracing on barrel (simulation): estimation of x- and y-position of the barrel (x_u^b and y_u^b). (by courtesy of Tinne De Laet and Wilm Decré)

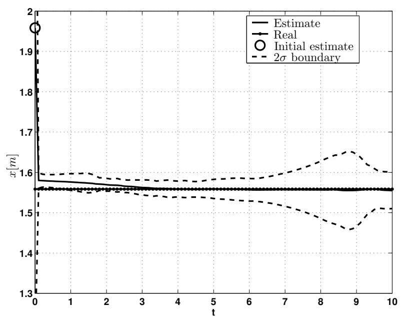
beneath the part. A force/torque sensor is attached to the robot wrist. The force/torque sensor allows the operator to interact with the robot by exerting forces on the machine part. Using these interaction forces, the operator aligns one side of the part with its desired location on the support. A camera provides information of the position of the other side of the part, relative to its desired position. These measurements are used by the robot to position that side of the part. Hence, task control is shared between the human and the robot. The robot carries the weight of the machine part, generates a downward motion to realize a contact force between the part and its support, and at the same time positions one side of the part based on the camera measurements. Simultaneously, the human aligns the other side of the part



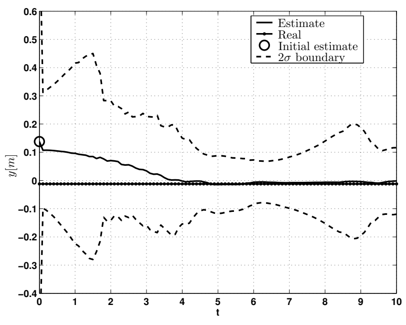
(a) Top view of the experiment.



(b) Laser distance measurement z_2.



(c) x-position of the barrel, x^b.



(d) y-position of the barrel, y^b.

Figure 6.13: Laser tracing on barrel (experiment): estimation of X and Y-position of the barrel. Note that the initial 2σ boundaries do not fit on the displayed area. (by courtesy of Tinne De Laet and Wilm Decré)

while maintaining overall control of the task.

The use of two sensors suggests two features: feature a for the visual servoing and feature b for the force control. For both features the relevant objects are the robot environment, $o1$, and the manipulated part, $o2$, as both the visual servoing and the force control concern motion of the part with respect to the environment. Figure 6.14 shows the object and feature frames (with a plate as manipulated part):

- Frame $o1$ is fixed to the robot environment. The camera frame is a possible choice for this frame, as the camera is fixed in the environment.
- Frame $o2$ is chosen at the center of the manipulated part.
- Frame $f1a$ is chosen at the reference pose on the support, to which the

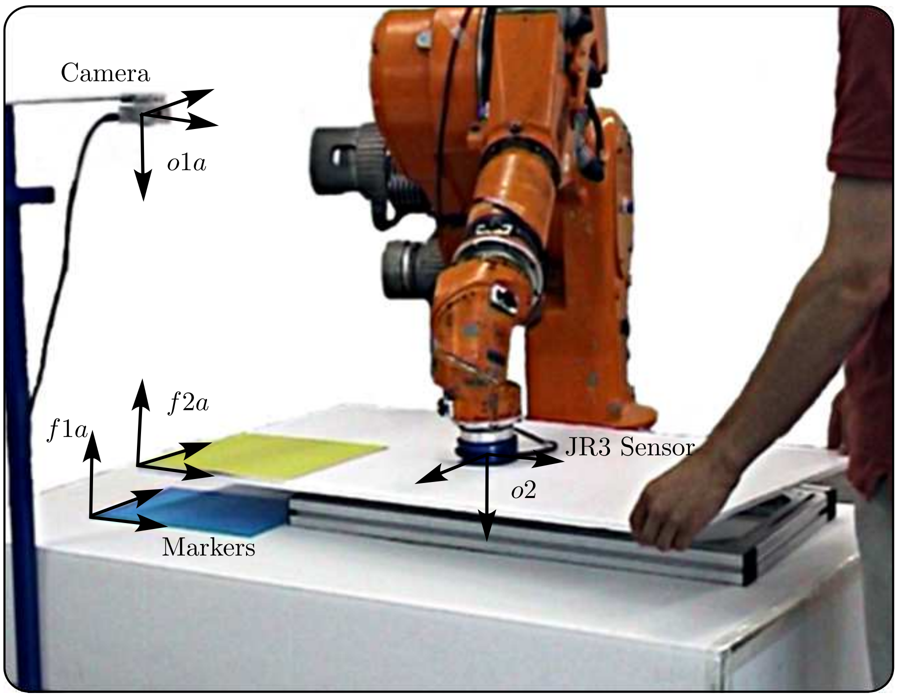


FIGURE 6.14: The object and feature frames for a human-robot co-manipulation task. In the experiment, the manipulated object is a plate. Colored sheets are attached to the plate and the table below it, to facilitate recognition of the object and the support in the camera images.

object should be aligned using the camera measurements. In the case of a fixed support, the position of $f1a$ with respect to $o1$ is known. If however the support is not fixed (for instance, if each time a different support is used and there is some variance on the exact location of the support), the position of $f1a$ is measured from the camera images.

- Frame $f2a$ is fixed to the object, at the reference point which should be aligned to $f1a$. While the pose of $f2a$ with respect to $o2$ is known, the relative pose between $f2a$ and $f1a$ is obtained from the camera measurements.

- Frame $f2b$ is attached to the manipulated part, at the reference position for the force control. That is, its origin is the reference point for the force control; measured torques will generate rotation around this point. A central point of the manipulated part is a common choice for the reference point. Hence, $f2b$ is chosen to coincide with $o2$.

- Frame $f1b$ is attached to the support, and is chosen to coincide with $f1a$.

For each feature the feature frames are fixed with respect to the object frames, and all six DOF between the objects are located between the feature frames. Hence similar feature Jacobians describe the motion of the frames for each feature. With $i = a$ or b:

$$\begin{aligned} \mathbf{t}_{o1}^{f1i} &= \boldsymbol{J}_{\boldsymbol{FI}}^{i} \boldsymbol{\tau}_{I}{}^{i}, \\ &= \mathbf{0}, \end{aligned} \tag{6.84}$$

$$\begin{aligned} {}^{f2i}_{f1i}\mathbf{t}_{f1i}^{f2i} &= \boldsymbol{J}_{\boldsymbol{FII}}^{i} \boldsymbol{\tau}_{II}{}^{i}, \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_1^i \\ \tau_2^i \\ \tau_3^i \\ \tau_4^i \\ \tau_5^i \\ \tau_6^i \end{bmatrix}, \end{aligned} \tag{6.85}$$

$$\begin{aligned} \mathbf{t}_{o2}^{f2i} &= \boldsymbol{J}_{\boldsymbol{FIII}}^{i} \boldsymbol{\tau}_{III}{}^{i}, \\ &= \mathbf{0}. \end{aligned} \tag{6.86}$$

The following constraints realize the desired behavior:

- The first two constraints are camera constraints, regulating the X and Y-motion of $f2a$ with respect to $f1a$ in the camera images[5]. Hence, these constraints contain the camera Jacobian (Section 4.3.3, page 61; with $\phi = 0$):

$$\begin{bmatrix} \frac{f}{z} & 0 & -\frac{fx}{z^2} \\ 0 & \frac{f}{z} & -\frac{fy}{z^2} \end{bmatrix} \begin{bmatrix} \tau_1^a \\ \tau_2^a \\ \tau_3^a \end{bmatrix} = k_{fb} \begin{bmatrix} x_{c,desired} - x_{c,measured} \\ y_{c,desired} - y_{c,measured} \end{bmatrix}. \tag{6.87}$$

  In this, $x_{c,measured}$ and $y_{c,measured}$ are the measured positions of $f2a$ in the camera image. $x_{c,desired}$ and $y_{c,desired}$ are the desired values for these positions (that is, the positions of $f1a$ in the camera image).

[5]Regulating the rotation in the camera image is also possible, but is not done in this example.

- The next three constraints are robot force constraints, and realize a contact force between the part and the support by regulating F_z, T_x and T_y, the force and torques along the Z, X and Y-axes respectively of $f1b$. The origin of $f2b$ is the reference point for the force control. Assuming that the contact stiffness between the part and the support can be described by a diagonal matrix diag$(k_x, k_y, k_z, k_{\alpha x}, k_{\alpha y}, k_{\alpha z})$, these constraints are given by:

$$\begin{aligned} \tau_3^b &= k_{fb}k_z^{-1}(F_{z,desired} - F_{z,measured}), && (6.88)\\ \tau_4^b &= k_{fb}k_{\alpha x}^{-1}(T_{x,desired} - T_{x,measured}), && (6.89)\\ \tau_5^b &= k_{fb}k_{\alpha y}^{-1}(T_{y,desired} - T_{y,measured}). && (6.90)\end{aligned}$$

  When there is no contact, these constraints result in a downwards motion, eventually resulting in contact between the part and its support.

- Three further force constraints regulate F_x, F_y and T_z, the forces and torque along the X, Y and Z-axes respectively, of $f1b$. These allow the operator to interact with the robot.

$$\begin{aligned} \tau_1^b &= k_{fb}k_x^{-1}(F_{x,desired} - F_{x,measured}), && (6.91)\\ \tau_2^b &= k_{fb}k_y^{-1}(F_{y,desired} - T_{y,measured}), && (6.92)\\ \tau_6^b &= k_{fb}k_{\alpha z}^{-1}(T_{z,desired} - T_{y,measured}). && (6.93)\end{aligned}$$

Constraints (6.91) and (6.92) conflict with the camera constraints (6.87), as they all control translational motion of the machine part in the horizontal plane. Hence, constraint weights have to be specified. As this weighting deliberates the error on the force relative to the error on the pose as measured by the camera, the weighted combination of the constraints acts like a spring: deviation from the camera position is possible by exerting force, but greater deviations necessite greater forces. The chosen weights influence the stiffness of this virtual spring. The greater the weights of the force constraints with respect to the position constraints, the easier it is to deviate from a desired position (as a deviation from the desired position is less important than a deviation from the desired force), and the lower the spring constant of this virtual spring will be.

This application has been carried out on a Kuka 361 industrial robot equipped with a wrist mounted JR3 force/torque sensor. Figure 6.14 shows the setup. The manipulated object consists of a rectangular plate, which is to be placed on a table. Colored sheets are attached to the plate and the table as markers, to facilitate recognition of the object and the support in the camera images. The application was implemented in Orocos (Bruyninckx 2001), and the visual recognition was done using LTI. Figure 6.15 shows some results.

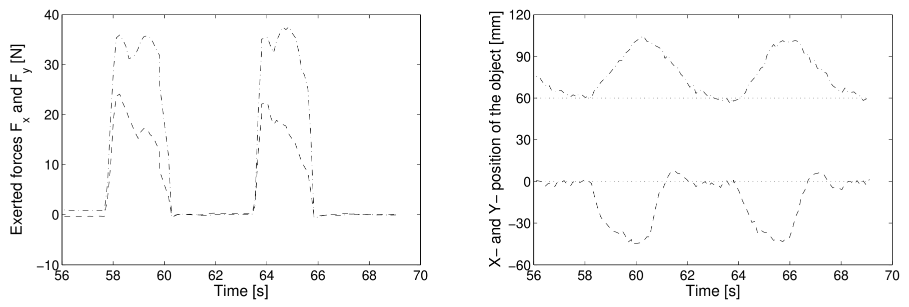


Figure 6.15: The left plot shows the forces F_x and F_y, exerted by the operator during the co-manipulation task. The right plot shows the alignment errors x^a and y^a as measured by the camera.

The left plot shows the F_x and F_y-forces, exerted by the operator. The right plot shows the alignment errors x^a and y^a, as measured by the camera. As the task is overconstrained, the forces exerted by the operator conflict with the aligning motion from the visual servoing, yielding a spring-like behavior. The relative weights of the force and visual servoing constraints determine the spring constant of this behavior.

6.8 Conclusions

In this chapter, iTASC is applied to several example applications, showing the effectiveness of the approach. Each of the applications illustrates different aspects of the task specification formalism. For instance, the minimally invasive surgery task with extra degrees of freedom shows that an existing task is **easily extensible** to a new setting, simply by changing parts of the model and adding constraints. The other examples deal with **over- and under-constrained** tasks, with **different kinds of robots** and **different sensors**, and illustrate the **model update procedure**, **estimation of geometric parameters**, and the **transition between states**.

Chapter 7

General Conclusions

During the last decades, research extensively focused on sensor-based robot tasks. Several task specification methodologies for these kinds of tasks were developed, such as the *Task Frame Formalism*, or the formalisms based on hybrid force/position control defined in terms of the wrench and twist spaces corresponding to a contact formation. However, most of these task specification formalisms exclusively focus on force control. Moreover, they cannot handle complex robot tasks, such as tasks dealing with multiple sensors concurrently or with general kinds of robots and constraints on multiple or intermediate links of those robots. Furthermore, these formalisms pay little to no attention to the estimation of geometric parameters.

The goal of our research is to fill this gap. We want to develop programming support for the implementation of complex, sensor-based robotic tasks in the presence of geometric uncertainty. The foundation for this programming support is iTASC: a generic and systematic approach to specify and control a sensor based task while dealing properly with geometric uncertainty.

The following two sections give an overview of the main contributions and conclusions of this thesis in the area of task specification for sensor-based tasks (Section 7.1), and an evaluation of the limitations of the formalism with suggestions for further research topics (Section 7.2).

7.1 Main contributions

This section lists the parts of this thesis that are original contributions to the state-of-the-art in robot task specification for sensor based tasks in the presence of geometric uncertainty.

Generic, constraint-based methodology to model and specify sensor based robot tasks

iTASC is a constraint-based methodology to specify sensor-based tasks. Inspired by the work of Ambler and Popplestone (1975), objects and features on these objects are introduced, and a task is formulated in terms of task relations between the features. However, where Ambler and Popplestone propose the use of analytic methods to solve the task relations, this text proposes to model the motion of the objects and features numerically in terms of feature Jacobians and feature twist coordinates. This allows us to express the task relations in terms of these auxiliary coordinates, which is straightforward as the feature twist coordinates are closely task related. Combined with the twist closure equations, the task relations are reformulated as a linear set of constraints on the joint velocities. This set of constraints corresponds to the first order expression of the Task Function (Samson, Le Borgne, and Espiau 1991), as needed for velocity resolved robots. The set of constraints can be both redundant or overconstrained; well-known approaches are used to solve the set of constraints numerically (Doty, Melchiorri, and Bonivento 1993; Nakamura 1991). In short, the specification formalism consists of the following four steps:

1. Choice of the object and features,
2. Modeling of the relative motion of the objects and features,
3. Definition of the constraints,
4. Solving for the instantaneous motion.

In contrast to previously presented approaches in which task specification is done in a single Task Frame or a single base, the methodology presented here is a true **constraint-based** approach. Constraints can be defined on any object-feature pair, so iTASC can cope with tasks involving multiple sources of constraints (such as multiple sensor inputs or constraints on different links of a robot). iTASC is a generic framework in that it does not focus on one particular kind of task, sensor or robot system: it can be used for **general, velocity resolved robot systems**, consisting of rigid links and joints. Also, where the previous approaches mostly focus on force control or, to a lesser extent, on the combination force-vision, in iTASC **all sensors yielding geometric information** can be used (for instance, cameras or distance sensors, but also dynamical sensors such as a force sensor, if the task is executed in a quasistatic way).

In Chapter 6 the methodology is applied to several example applications, showing the effectiveness of the approach. Each of the applications illustrates different aspects of the task specification formalism. For instance, the minimally invasive surgery task with extra degrees of freedom shows that an existing task is **easily extensible** to a new setting, by changing parts of the model and adding constraints. The other examples deal with **over- and underconstrained** tasks, with different kinds of robots and different sensors, and illustrate the **model update procedure** and the **estimation of geometric parameters**.

Model update procedure

In the task specification methodology, objects and features are represented by frames. At each timestep, the pose of these frames must be known. This requirement is also present in other specification approaches, where for example the pose of a Task Frame or an instantaneous twist and wrench base is needed. With the exception of for instance the *track-on-force* and *track-on-velocity* methods in COMRADE, task specification methodologies generally assume that the Task Frames or bases are known or can be calculated, for instance from an (analytic) description of the object geometries. In contrast to this, Section 5.2 presents a generic update procedure to keep track of the frame poses. While in most approaches the programmer needs to implement ad-hoc procedures to calculate these poses, the generic update procedure automatically tracks the poses of the object and feature frames. Furthermore, such a procedure is a necessary component of future software support for the task specification, which will hide the mathematics and inner workings of the task specification procedure as much as possible from the end user.

The model update procedure consists of a prediction and a correction step. In the prediction step, the object and feature frame twists are integrated, to obtain a prediction of the new object and feature frame poses. Iterative application of this prediction is prone to accumulation of integration errors due to discretization and due to errors in the geometric model. Therefore the predicted poses are corrected in the correction step, based on the additional information contained in the pose closure equations.

Estimation of geometric parameters

Section 5.3 integrates the estimation of geometric parameters into the specification framework. To this end, an additional set of uncertainty coordinates which model the unknown geometric parameters is introduced, in a similar way as the feature twist coordinates. The measurement models, relating the measurements to the geometric parameters, are expressed in terms of the feature twist coordinates. For well chosen object and feature frames, it is very straightforward to define these measurement models in terms of the feature twist coordinates. Finally, the twist closure equations are used as extra measurement equations, relating the feature twist coordinates to the uncertainty coordinates. The procedures proposed here are generic, in that they allow identification of any kind of geometric parameter which is observable, and that they do not focus on a specific identification problem, such as the tracking of a Task Frame. To the author's knowledge no previous attempts have been made to integrate the estimation of geometric parameters into the task specification process in such a generic way.

7.2 Limitations and future work

This section describes the limitations of iTASC and suggests topics for further research.

Fields of application

The ultimate goal of sensor-based robotics is to create intelligent machines, capable of performing tasks autonomously in unstructured environments. The state of the art, including iTASC, is still far away from this target. This section discusses the fields of application of iTASC, both with respect to the target users, as well as to the target environments.

Target users iTASC is a methodology for low-level task specification of sensor-based robot tasks. It has a strong focus on geometric modeling and hence primarily targets *system integrators*, who develop applications for end-users. System integrators can learn the principles of iTASC, and gain the insight needed to apply these to specify different tasks. Insight is still needed to specify a task, as there are no *strict* rules to choose for instance the object and feature frames or the feature Jacobians. A good choice is necessary, as it is difficult to express the constraints in terms of ill-chosen feature twist coordinates. This is clearly a disadvantage, but it is not something peculiar to iTASC: similar insight is also required in other task specification methodologies, notwithstanding their successful application for multiple tasks. For instance, also according to the TFF it is difficult to specify a task if the Task Frame is ill-chosen. This inspired the publication of reference papers such as (Bruyninckx and De Schutter 1996), in which multiple examples are given of tasks specified according to the TFF. Similarly, in Chapter 6, this thesis gives multiple examples of tasks specified according to iTASC.

Furthermore, software is being developed at our research group to support system integrators in specifying a task. In this software, the motion degrees of freedom between the object and feature frames are specified as links of a kinematic chain. The feature Jacobians are automatically deduced from this model, and the mathematical representations are hidden from the integrator.

For *end-users* of an applications, the iTASC methodology is usually completely hidden. To interact with the task, they use a task-specific interface which is provided by the system integrator. In general, such an interface allows the end-user to change parameters of the task, such as the control gains or the active state, but hides the internal model, such as the feature Jacobians, from the user.

Target environments In iTASC, the programmer specifies a task by defining constraints and estimators, based on a geometric model of the task. To foresee the different states of the task and to specify the different models and constraints for each state, the environment should have a certain degree of structure. This is for instance the case in industrial environments, in which the objects are known, but not necessarily their exact locations in the robot environment. In other words, iTASC primarily targets *semi-structured* environments.

When using robots in a *completely unstructured* environment, such as a domestic environment, it is impossible to implement different constraints and estimators for all possible situations that can occur. In such settings, higher-level artificial intelligence is needed, capable of making autonomous, strategic decisions. The development of such a cognitive system is an interesting research topic, and has been an active research topic for the last decades. Nevertheless, the state-of-the-art is still far away from what can be called a true intelligent system. In our research group, modeling techniques are being developed to build geometric models of an unknown environment based on contact measurements (force and position) (Slaets et al. 2007). A possible topic for further research is to integrate these techniques into iTASC. A given target action between two objects can then be executed once these objects have been found in the robot environment.

Another possibility to use robots in unstructured environments is to have the robot and the operator work together to realize a task. For instance, an operator uses a teleoperation setup to move a robot within the range of target objects in its environment, and once these are identified, the robot autonomously continues the task.

In any case, iTASC can be used as the underlying methodology for these tasks, once the robot environment has been identified to such an extent that a geometric model of the task can be built and constraints specified within this model.

Framework for instantaneous control

iTASC is a framework for instantaneous control. The generated models are instantaneous kinematic models of the task, and are used to generate velocity setpoints, that is, instantaneous joint velocities. This implies that path planning is limited to trajectory generation, for instance using trapezoidal or minimum-jerk trajectories. Also the optimization criteria in iTASC are local criteria, minimizing an instantaneous norm of the joint velocities or of the constraint violation. The extension of iTASC to incorporate global path planning, capable of error recovery and re-planning, and global optimization, is a challenging research topic.

Focus on velocity resolved robots

According to iTASC, a task is specified in such a way that a first order expression of the Task Function is obtained, as needed for velocity resolved robots. For acceleration (or torque) resolved robots, a second order expression is needed (De Laet and De Schutter 2007). The main reasons that this thesis focuses on velocity resolved systems are (i) that many tasks can be adequately specified on a velocity level, (ii) that many robots which are interfaced to an external control computer are effectively velocity resolved systems, and (iii) that the mathematical expressions for acceleration resolved systems are more involved, as it requires expressions for the time derivatives of the feature Jacobians. It is much harder to gain insight in and specify feature Jacobians at the acceleration level. These are of course no decisive reasons: nothing prohibits the formulation of the task specification formalism on an acceleration level. However, because of the added complexity of specifying the feature Jacobians at the acceleration level, it is the author's belief that in this case software support is definitely needed for the formalism to be practically usable.

Inequality constraints

iTASC is based on the definition of equality constraints to realize a task. For some applications however, it is desirable to specify inequality constraints. Such constraints express a *don't care* behavior as long as the inequality is not violated. For instance, consider a robot of which some links are close to a wall. For those links it is important that the motion is restricted to avoid collision, while such a restriction is not needed for the other links. A possible way to realize such inequality constraints could be to add extra equality constraints, with a weighting which differs from zero only in the neighborhood of the inequality violation. In the example above, these constraints would realize a *move away from wall* behavior, and have zero weights for all but the links close to the wall.

Non-linear estimation techniques

The estimation techniques described in Section 5.3 are based on a linearization of the measurement equations, as for instance needed for a Kalman Filter. However, a linearized approach is not always appropriate for strongly non-linear estimation problems. In such cases the use of different techniques, such as a sequential Monte Carlo filter (particle filter) can yield a better result.

References

ABB (2006). ABB Robotics. `http://www.abb.com/robotics/`.

Aghili, F. (2005). A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: applications to control and simulation. *IEEE Transactions on Robotics 21*(5), 834–849.

Ahmad, S. and S. Luo (1989). Coordinated motion control of multiple robotic devices for welding and redundancy coordination through constrained optimization in Cartesian space. *IEEE Transactions on Robotics and Automation 5*(4), 409–417.

Aksenov, G. S., D. K. Voronetskaya, and V. N. Fomin (1978). A construction of program movements of a manipulator with the aid of a computer. *Engineering Cybernetics 16*(4), 40–45.

Ambler, A. P. and R. J. Popplestone (1975). Inferring the positions of bodies from specified spatial relationships. *Artificial Intelligence 6*, 157–174.

An, C. H. and J. Hollerbach (1989). The role of dynamic models in Cartesian force control of manipulators. *IJRR 8*(4), 51–72.

Anderson, R. J. and M. W. Spong (1988). Hybrid impedance control of robotics manipulators. *IEEE Journal of Robotics and Automation 4*(5), 549–556.

Baeten, J., H. Bruyninckx, and J. De Schutter (2003). Integrated vision/force robotics servoing in the task frame formalism. *The International Journal of Robotics Research 22*(10), 941–954.

Baeten, J. and J. De Schutter (2003). *Integrated Visual Servoing and Force Control. The Task Frame Approach.* Springer Tracts in Advanced Robotics. Springer Verlag.

Ben-Israel, A. and T. N. E. Greville (1980). *Generalized Inverses: Theory and Applications* (Reprinted ed.). Huntington, NY: Robert E. Krieger Publishing Company.

Blajer, W. (1997). A geometric unification of constrained system dynamics. *Multibody System Dynamics 1*(1), 3–21.

Bonaventura, C. S. and K. W. Jablokow (2005). A modular approach to the dynamics of complex multirobot systems. *IEEE Transactions on Robotics 21*(1), 26– 37.

Bredereke, J. and A. Lankenau (2005). Safety-relevant mode confusions – modelling and reducing them. *Reliability Engineering and System Safety 88*(3), 229–245.

Bruemmer, D. J., D. A. Few, R. L. Boring, J. L. Marble, M. C. Walton, and C. W. Nielsen (2005). Shared understanding for collaborative control. *IEEE Transactions on Systems, Man, and Cybernetics. Part A: Systems and Humans 35*(4), 494–504.

Bruyninckx, H. (2001). Open RObot COntrol Software. `http://www.orocos.org/`.

Bruyninckx, H. and J. De Schutter (1996). Specification of force-controlled actions in the "Task Frame Formalism": A survey. *IEEE Transactions on Robotics and Automation 12*(5), 581–589.

Bruyninckx, H. and J. De Schutter (1997). Where does the Task Frame go? In Y. Shirai and S. Hiroshe (Eds.), *Robotics Research, The Eight International Symposium*, Shonan, Japan, pp. 55–65. Springer-Verlag.

Bruyninckx, H. and O. Khatib (2000). Gauss' Principle and the dynamics of redundant and constrained manipulators. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp. 2563–2568.

Caccavale, F., C. Natale, B. Siciliano, and L. Villani (2005). Integration for the next generation: embedding force control into industrial robots. *IEEE Robotics and Automation Magazine 12*(3), 53–64.

Canudas de Wit, C., B. Siciliano, and G. Bastin (Eds.) (1996). *Theory of Robot Control*. Communications and Control Engineering. London, England: Springer.

Chang, K.-S. and O. Khatib (2000). Operational space dynamics: efficient algorithms for modeling and control of branching mechanisms. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp. 850–856.

Chen, J. and A. Zelinsky (2003). Programing by demonstration: Coping with suboptimal teaching actions. *The International Journal of Robotics Research 22*(5), 299–319.

Chiaverini, S. and L. Sciavicco (1993). The parallel approach to force/position control manipulators. *IEEE Transactions on Robotics and Automation 9*, 289–293.

Cortesão, R., J. Park, and O. Khatib (2003a). Real-time adaptive control for haptic manipulation with active observers. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, USA, pp. 2938–2943.

Cortesão, R., J. Park, and O. Khatib (2003b). Robust and adaptive teleoperation for compliant motion tasks. In *Proceedings of the 2003 International Conference on Advanced Robotics*, Coimbra, Portugal, pp. 513–519.

Critchley, J. and K. S. Anderson (2003). A generalized recursive coordinate reduction method for multibody system dynamics. *International Journal for Multiscale Computational Engineering 1*(2–3), 181–199.

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the 9th International Conference on Computer Vision*, Nice, France, pp. 1403–1410.

de Jalón, J. G. and E. Bayo (1993). *Kinematic and Dynamic Simulation of Multibody Systems—The Real Time Challenge*. Springer.

De Laet, T. and J. De Schutter (2007). Control schemes for constraint-based task specification in the presence of geometric uncertainty using auxiliary coordinates. Internal report 07RP001, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium.

De Schutter, J. (1988). Improved force control laws for advanced tracking applications. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, Philadelphia, PA, pp. 1497–1502.

De Schutter, J. and H. Bruyninckx (1996). Force control of robot manipulators. In *The Control Handbook*, pp. 1351–1358. CRC Press.

De Schutter, J., H. Bruyninckx, S. Dutré, J. De Geeter, J. Katupitiya, S. Demey, and T. Lefebvre (1999, Dec). Estimating first-order geometric parameters and monitoring contact transitions during force-controlled compliant motions. *The International Journal of Robotics Research 18*(12), 1161–1184.

De Schutter, J., H. Bruyninckx, W.-H. Zhu, and M. W. Spong (1997). Force control: a bird's eye view. In B. Siciliano (Ed.), *Control Problems in Robotics and Automation: Future Directions*, pp. 1–17. San Diego, CA: Springer.

De Schutter, J. and J. Leysen (1987). Tracking in compliant robot motion: Automatic generation of the task frame trajectory based on observation of the natural constraints. In R. Bolles (Ed.), *Proceedings of the 4th Int. Symposium of Robotics Research*, Santa Cruz, CA. MIT Press.

De Schutter, J., J. Rutgeerts, E. Aertbelien, F. De Groote, T. De Laet, T. Lefebvre, W. Verdonck, and H. Bruyninckx (2005). Unified constraint-based task specification for complex sensor-based robot systems. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, pp. 3618–3623.

De Schutter, J., D. Torfs, S. Dutré, and H. Bruyninckx (1997). Invariant hybrid force/position control of a velocity controlled robot with compliant end effector using modal decoupling. *The International Journal of Robotics Research 16*(3), 340–356.

De Schutter, J. and H. Van Brussel (1988a, Aug). Compliant Motion I, II. *The International Journal of Robotics Research 7*(4), 3–33.

De Schutter, J. and H. Van Brussel (1988b). Compliant robot motion I. A formalism for specifying compliant motion tasks. *The International Journal of Robotics Research 7*(4), 3–17.

De Schutter, J. and H. Van Brussel (1988c). Compliant robot motion II. A control approach based on external control loops. *The International Journal of Robotics Research 7*(4), 18–33.

Delmia (2006). Delmia v5 robotics. `http://www.delmia.com/gallery/pdf/DELMA_V5Robotics.pdf`.

Demeester, E., M. Nuttin, D. Vanhooydonck, and H. Van Brussel (2003, March). Assessing the user's intent using Bayes' rule: Application to wheelchair control. In *Proc. of ASER 2003*, Bardolino, Italy, pp. 117–124.

Doty, K. L., C. Melchiorri, and C. Bonivento (1993). A theory of generalized inverses applied to robotics. *The International Journal of Robotics Research 12*(1), 1–19.

Doucet, A., N. J. Gordon, and V. Krishnamurthy (2001, march). Particle Filters for State Estimation of Jump Markov Linear Systems. *IEEE Transactions on Signal Processing 49*(3), 613–624.

Duffy, J. (1990). The fallacy of modern hybrid control theory that is based on "orthogonal complements" of twist and wrench spaces. *Journal of Robotic Systems 7*(2), 139–144.

English, J. D. and A. A. Maciejewski (2000). On the implementation of velocity control for kinematically redundant manipulators. *ITSMC 30*(3), 233–237.

Espiau, B., F. Chaumette, and P. Rives (1992). A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation 8*(3), 313–326.

Estrada, C., J. Neira, and J. D. Tardós (2005). Hierarchical SLAM: real-time accurate mapping of large environments. *IEEE Transactions on Robotics 21*(4), 588–596.

Featherstone, R. (2004). Modeling and control of contact between constrained rigid bodies. *The International Journal of Robotics Research 23*(1), 82–92.

Featherstone, R., S. Sonck, and O. Khatib (1999). A general contact model for dynamically-decoupled force/motion control. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, Detroit, MI, pp. 3281–3286.

Fernandez, V., C. Balaguer, D. Blanco, and M. A. Salichs (2001). Active human-mobile manipulator cooperation through intention recognition. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea, pp. 2668–2673.

Ferretti, G., G. Magnani, and P. Rocco (2004). Impedance control for elastic joints industrial manipulators. *IEEE Transactions on Robotics and Automation 20*(3), 488–498.

Fisher, W. D. and M. S. Mujtaba (1992). Hybrid position/force control: A correct formulation. *The International Journal of Robotics Research 11*(4), 299–311.

Fruchard, M., P. Morin, and C. Samson (2006). A framework for the control of nonholonomic mobile manipulators. *The International Journal of Robotics Research 25*(8), 745–780.

Garćia, J. G., A. Robertsson, J. G. Ortega, and R. Johansson (2005). Force and acceleration sensor fusion for compliant robot motion control. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, pp. 2709–2714.

Garćia, J. G., A. Robertsson, J. G. Ortega, and R. Johansson (2006). Generalized contact force estimation for a robot manipulator. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, U.S.A., pp. 4019–4024.

Gautier, M. (1986). Identification of robot dynamics. In *Proceedings of the IFAC Symposium on Theory of Robots*, Vienna, Austria, pp. 351–356.

Glover, J., S. Thrun, and J. T. Matthews (2004). Learning user models of mobility-related activities through instrumented walking aids. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, U.S.A., pp. 3306–3312.

Hogan, N. (1985). Impedance control: An approach to manipulation. Parts I-III. *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control 107*, 1–24.

Hogan, N. (1987). Stable execution of contact tasks using impedance control. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, Raleigh, NC, pp. 1047–1054.

Huang, H.-k. and G. C. I. Lin (2003). Rapid and flexible prototyping through a dual-robot workcell. *Robotics and Computer Integrated Manufacturing 19*(3), 263–272.

Jankowski, K. P. and H. A. ElMaraghy (1996). Constraint formulation for invariant hybrid position/force control of robots. *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control 118*, 290–299.

Jouaneh, M. K., D. A. Dornfeld, and M. Tomizuka (1990). Trajectory planning for coordinated motion of a robot and a positioning table: Part 2—optimal trajectory specification. *IEEE Transactions on Robotics and Automation 6*(6), 746–759.

Jouaneh, M. K., Z. Wang, and D. A. Dornfeld (1990). Trajectory planning for coordinated motion of a robot and a positioning table: Part 1—path specification. *IEEE Transactions on Robotics and Automation 6*(6), 735–745.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering 82*, 34–45.

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

Khatib, O., O. Brock, K.-S. Chang, D. Ruspini, L. Sentis, and S. Viji (2003). Robots for the human and interactive simulations. In T. Huang (Ed.), *Proceedings of the 11th World Congress in Mechanism and Machine Science*, Tianjin, China, pp. 1572–1576. China Machinery Press.

Khatib, O., O. Brock, K.-S. Chang, D. Ruspini, L. Sentis, and S. Viji (2004). Human-centered robotics and interactive haptic simulation. *The International Journal of Robotics Research 23*(2), 167–178.

Klein, C. A. and C. H. Huang (1983). Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Transactions on Systems, Man, and Cybernetics 13*, 245–250.

Kröger, T., B. Finkemeyer, M. Heuck, and F. M. Wahl (2004). Adaptive implicit hybrid force/pose control of industrial manipulators: Compliant motion experiments. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, pp. 816–821.

Kröger, T., B. Finkemeyer, and F. M. Wahl (2004). A task frame formalism for practical implementations. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, U.S.A., pp. 5218–5223.

Kröger, T., D. Kubus, and F. M. Wahl (2006). 6D force and acceleration sensor fusion for compliant motion control. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, pp. 2626–2630.

Kulić, D. and E. Croft (2003). Estimating intent for human robot interaction. In *Proceedings of the 2003 International Conference on Advanced Robotics*, Coimbra, Portugal, pp. 810 – 815.

Latombe, J. C. (1989). Motion planning with uncertainty: on the preimage backchaining approach. In O. Khatib, J. J. Craig, and T. Lozano-Pérez (Eds.), *The Robotics Review*, pp. 55–69. MIT Press.

Latombe, J. C. (1991). *Robot motion planning*, Volume 124 of *Int. Series in Engineering and Computer Science*. Boston, MA: Kluwer Academic Publishers.

Latombe, J. C. (1999). Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *The International Journal of Robotics Research 18*(11), 1119– 1128. Invited paper.

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

Lefebvre, T., H. Bruyninckx, and J. De Schutter (2005a). *Nonlinear Kalman Filtering for Force-Controlled Robot Tasks*. Springer Tracts in Advanced Robotics. Springer Verlag.

Lefebvre, T., H. Bruyninckx, and J. De Schutter (2005b). Online statistical model recognition and state estimation for autonomous compliant motion. *IEEE Transactions on Systems, Man, and Cybernetics. Part C: Applications and Reviews 35*(1), 16–29.

Leonard, J. J. and H. F. Durrant-Whyte (1991). Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings of the 1991 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Osaka, Japan, pp. 1442–1447.

Leonard, J. J. and H. F. Durrant-Whyte (1992). *Directed sonar sensing for mobile robot navigation*. Boston, MA: Kluwer Academic Publishers.

Liegeois, A. (1977). Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics SMC-7*(12), 868–871.

Lipkin, H. and J. Duffy (1988). Hybrid twist and wrench control for a robotic manipulator. *Transactions of the ASME, Journal of Mechanisms, Transmissions, and Automation in Design 110*, 138–144.

Liu, G., K. Dubowsky, and G. Morel (1998). A base force/torque sensor approach to robot manipulator inertial parameter estimation. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, Leuven, Belgium, pp. 3316–3321.

Liu, G. and Z. Li (2002). A unified geometric approach to modeling and control of constrained mechanical systems. *IEEE Transactions on Robotics and Automation 18*(4), 574–587.

Liu, Y.-H., Y. Xu, and M. Bergerman (1999). Cooperation control of multiple manipulators with passive joints. *IEEE Transactions on Robotics and Automation 15*(2), 258–267.

Lozano-Pérez, T. (1983). Spatial planning: A configuration space approach. *IEEE Transactions on Computers C-32*(2), 108–120.

Marcelo, H., W. Lin, and S.-Y. Lim (1999). A walk-through programmed robot for welding in shipyards. *The Industrial Robot 26*(5), 377–388.

Mason, M. and K. Salisbury (1985). *Robot Hands and the Mechanics of Manipulation.* MIT Press.

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

Metris (2007). Metris. `http://www.metris.com/`.

Miller, D. J. and R. C. Lennox (1990). An object-oriented environment for robot system architectures. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, OH, pp. 352–361.

Montemayor, G. and J. T. Wen (2005). Decentralized collaborative load transport by multiple robots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, pp. 372–377.

Murray, R. M., Z. Li, and S. S. Sastry (1994). *A mathematical introduction to robotic manipulation.* Boca Raton, FL: CRC Press.

Mustapic, G., J. Andersson, C. Norström, and A. Wall (2004). A dependable open platform for industrial robotics: A case study. In *Architecting*

Dependable Systems II, Lecture Notes in Computer Science, pp. 307–329. Springer.

Nakamura, Y. (1991). *Advanced robotics: redundancy and optimization.* Reading, MA: Addison-Wesley.

Nakamura, Y., H. Hanafusa, and T. Yoshikawa (1987). Task-priority based redundancy control of robot manipulators. *The International Journal of Robotics Research 6*(2), 3–15.

Natale, C. (2003). *Interaction control of robot manipulators – Six-Degrees-of-Freedom tasks*, Volume 3 of *Springer Tracts in Advanced Robotics.* London, UK: Springer-Verlag.

Owen, W. S., E. A. Croft, and B. Benhabib (2005). Acceleration and torque redistribution for a dual-manipulator system. *IEEE Transactions on Robotics 21*(6), 1226–1230.

Park, J. and O. Khatib (2005). Multi-link multi-contact force control for manipulators. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, pp. 3624–3629.

Penrose, R. (1955). A generalized inverse for matrices. *Proc. Cambridge Philos. Soc. 51*, 406–413.

Popplestone, R. J., R. Weiss, and Y. Liu (1988). Using characteristic invariants to infer new spatial relationships. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, Philadelphia, PA, pp. 1107–1112.

Raibert, M. and J. J. Craig (1981). Hybrid position/force control of manipulators. *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control 102*, 126–133.

Ruspini, D. and O. Khatib (1999). Collision/contact models for dynamic simulation and haptic interaction. In J. Hollerbach and D. Koditschek (Eds.), *Robotics Research, The Ninth International Symposium*, Snowbird, Utah, pp. 185–194. Springer-Verlag.

Salisbury, J. K. (1980). Active stiffness control of a manipulator in Cartesian coordinates. In *19th IEEE Conf. on Decision and Control.*

Samson, C., M. Le Borgne, and B. Espiau (1991). *Robot Control, the Task Function Approach.* Oxford, England: Clarendon Press.

Saridis, G. N. (1979). *Self-organizing control of stochastic systems.* New York, NY: Marcel Dekker.

Siciliano, B. (1995). Parallel force/position control of robot manipulators. In G. Giralt and G. Hirzinger (Eds.), *Robotics Research, the 7th International Symposium*, London, UK, pp. 78–89. Springer-Verlag.

Siciliano, B. and L. Villani (1999). *Robot Force Control.* Kluwer Academic Publishers.

Sirouspour, S. (2005). Modelling and control of cooperative teleoperation systems. *IEEE Transactions on Robotics 21*(6), 1220–1225.

Slaets, P., T. Lefebvre, J. Rutgeerts, H. Bruyninckx, and J. De Schutter (2007). Incremental building of a polyhedral feature model for programming by human demonstration of force controlled tasks. *IEEE Transactions on Robotics 23*(1), 20–33.

Sminchisescu, C., D. Metaxas, and S. Dickinson (2005). Incremental model-based estimation using geometric constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*(3), 727–738.

Soatto, S., R. Frezza, and P. Perona (1996). Motion estimation via dynamic vision. *IEEE Transactions on Automatic Control 41*, 393–414.

Sorenson, H. W. (1970). Least-squares estimation from Gauss to Kalman. *IEEE Spectrum 7*, 63–68.

Sorenson, H. W. (1985). *Kalman filtering: theory and application.* New York, NY: IEEE Press.

Stäubli (2006). Stäubli Robotics. `http://www.staubli.com/web/robot/division.nsf`.

Sugihara, T. and Y. Nakamura (2002). Whole-body cooperative balancing of humanoid robot using COG jacobian. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, pp. 2563–2568.

Swevers, J., C. Ganseman, J. De Schutter, and H. Van Brussel (1996). Experimental robot identification using optimised periodic trajectories. *Mechanical Systems and Signal Processing 10*(5), 561–577.

Tanizaki, H. (1996). *Nonlinear Filters. Estimation and Applications.* Springer-Verlag.

Taylor, R. H. (1976). *Synthesis of manipulator control programs from task-level specifications.* Ph. D. thesis, Department of Computer Science, Stanford University, Stanford CA.

Todd, D. (1986). *Fundamentals of Robot Technology.* New York, NY: John Wiley & Sons.

Van de Poel, P., J. De Schutter, and H. Van Brussel (1994). Robotise polishing of precast concrete surfaces using sensor control: feasibility study report. Internal report 94R31, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium.

Van de Poel, P., W. Witvrouw, H. Bruyninckx, and J. De Schutter (1993). An environment for developing and optimizing compliant robot motion tasks. In *Proceedings of the 1993 International Conference on Advanced Robotics*, Tokyo, Japan, pp. 713–718.

Verdonck, W., J. Swevers, and J.-C. Samin (2001). Experimental dynamic robot identification: advantages of combining internal and external measurements and of using periodic excitation. *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control 123*(4), 630–636.

Wang, Q. (1999, November). *Programming of compliant robot motion by human demonstration*. Ph. D. thesis, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium.

Whitney, D. E. (1977). Force feedback control of manipulator fine motions. *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control 99*(2), 91–97.

Whitney, D. E. (1987). Historical perspective and state of the art in robot force control. *The International Journal of Robotics Research 6*(1), 3–14.

Witvrouw, W. (1996). *Development of experiments and environment for sensor controlled robot tasks*. Ph. D. thesis, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Belgium.

Witvrouw, W., P. Van de Poel, and J. De Schutter (1995). Comrade: Compliant motion research and development environment. In *3rd IFAC/IFIP workshop on Algorithms and Architectures for Real-Time Control*, Ostend, Belgium, pp. 81–87.

Wu, L., K. Cui, and S. B. Chen (2000). Redundancy coordination of multiple robotic devices for welding through genetic algorithm. *Robotica 18*(6), 669–676.

Yoshikawa, T. (2000). Force control of robot manipulators. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp. 220–226.

Yoshikawa, T., T. Sugie, and N. Tanaka (1988). Dynamic hybrid position/force control of robot manipulators—controller design and experiments. *IEEE Journal of Robotics and Automation 4*(6), 699–705.

Index

Curriculum Vitae

Personal data

Johan Rutgeerts
27 October 1979, Roeselare, Belgium
johan.rutgeerts@gmx.net

Education

- **2002-2007**: **Ph.D. in mechanical engineering** at the Katholieke Universiteit Leuven, Belgium.

  My research is situated in the area of task specification for sensor-based robot systems in the presence of geometrical uncertainty. The aim of this research is to develop a task specification methodology, which allows users to specify complex robot tasks incorporating estimation of geometrical parameters in a generic and systematic way.

- **2000 - 2002**: **Master of science in mechanical engineering** specialization mechatronics and machine design at the Katholieke Universiteit Leuven, Belgium.

  2000 - 2001: **ECTS** (European Credit Transfer System) one year study exchange with the Graz University of Technology, Austria.

  2001: **Athens** program at the Ecole Nationale des Ponts et Chaussées, Paris, France.

List of Publications

Bruyninckx, H., J. De Schutter, T. Lefebvre, K. Gadeyne, P. Soetens, J. Rutgeerts, P. Slaets, and W. Meeussen (2003). Building blocks for slam in autonomous compliant motion. In R. Chatila, P. Dario, and O. Khatib (Eds.), *Robotics Research, the 11th International Symposium*, Siena, Italy, pp. 432–441.

De Schutter, J., T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx (2007). Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *The International Journal of Robotics Research*. To appear in May 2007.

De Schutter, J., J. Rutgeerts, E. Aertbelien, F. De Groote, T. De Laet, T. Lefebvre, W. Verdonck, and H. Bruyninckx (2005). Unified constraint-based task specification for complex sensor-based robot systems. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, pp. 3618–3623.

Meeussen, W., J. Rutgeerts, K. Gadeyne, H. Bruyninckx, and J. De Schutter (2006a). Bayesian contact state segmentation for programming by human demonstration in compliant motion tasks. In *Proceedings of the International Symposium on Experimental Robotics*, Rio de Janeiro, Brazil. in press.

Meeussen, W., J. Rutgeerts, K. Gadeyne, H. Bruyninckx, and J. De Schutter (2006b). Contact state segmentation using particle filters for programming by human demonstration in compliant motion tasks. *IEEE Transactions on Robotics*. in press.

Meeussen, W., J. Rutgeerts, K. Gadeyne, H. Bruyninckx, and J. De Schutter (2006c). Particle filters for hybrid event sensor fusion with 3d vision and force. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Heidelberg, Germany, pp. 518–523.

Rutgeerts, J., P. Slaets, F. Schillebeeckx, W. Meeussen, B. Stallaert, P. Princen, T. Lefebvre, H. Bruyninckx, and J. De Schutter (2005).

A demonstration tool with Kalman Filter data processing for robot programming by human demonstration. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Canada, pp. 3918–3923.

Slaets, P., J. Rutgeerts, K. Gadeyne, T. Lefebvre, H. Bruyninckx, and J. De Schutter (2004). Construction of a Geometric 3-D Model from Sensor Measurements Collected during Compliant Motion. In *Proceedings of the International Symposium on Experimental Robotics*, Singapore, Australia, pp. 571–580. in press.

Nederlandse Samenvatting

Taakspecificatie met behulp van beperkingen en schatting van geometrische onzekerheden voor sensorgebaseerde robottoepassingen

Nederlandse Samenvatting

1 Inleiding

Vrijwel alle robots die op dit moment in gebruik zijn in een niet-experimentele omgeving, zijn blinde machines die zich niet gewaar zijn van hun omgeving. De taken die een dergelijke robot uitvoert zijn pure positioneringstaken, zoals trajectvolgen of *pick-and-place*-toepassingen. Deze robottaken bestaan uit het afspelen van een voorgeprogrammeerd traject, waarbij op gezette tijden een actie uitgevoerd wordt zoals het spuiten van verf of het plaatsen van een puntlas. Aangezien de robot geen informatie heeft over de toestand van zijn omgeving tijdens de uitvoering van de taak, moet de omgeving volledig overeenkomen met wat vooropgesteld is bij het programmeren van de taak. De omgeving van de robot moet dus volledig gestructureerd zijn, en aangepast zijn aan de robot. Dit brengt aanzienlijke kosten met zich mee, en zorgt ervoor dat het gebruik van robots voornamelijk beperkt blijft tot taken die veelvuldig herhaald moeten worden, zoals assemblagetaken bij massaproductie.

Een mogelijke oplossing om robots in te zetten in minder gestructureerde omgevingen, is het gebruik van sensoren. Door een robot uit te rusten met extra sensoren, zoals een krachtsensor, een camera of een afstandssensor, kan de robot zijn omgeving waarnemen, en de taakuitvoering aanpassen aan de toestand van de omgeving.

Sensorgebaseerde robotica kan slechts veelvuldig toegepast worden, als sensorgebaseerde taken eenvoudig gespecificeerd kunnen worden. Om de taakspecificatie voor dergelijke toepassingen te ondersteunen werden in onderzoeksomgevingen verschillende methodologieën ontwikkeld, zoals het *Task Frame Formalism* (TFF) (De Schutter and Leysen 1987), of een aantal formalismen gebaseerd op hybride kracht/positie-controle (Aghili 2005; Liu and Li 2002).

Hoewel deze methodologieën zeer nuttig zijn om bepaalde soorten toepassingen te specificeren, hebben ze een aantal gemeenschappelijke nadelen. Ze gebruiken eenvoudige geometrische modellen om de taakspecificatie te ondersteunen, zoals één enkel taakassenstelsel of één enkele snelheids- en krachtbasis. Dit betekent dat dergelijke methodologieën ongeschikt zijn om taken te specificeren die verschillende bronnen van controlebeperkingen vereisen, zoals taken waarbij verschillende sensoren gedeeltelijke informatie leveren om de taak te realiseren. Ook bijvoorbeeld taken waarbij er deelspecificaties zijn voor verschillende delen van de robot, kunnen niet of moeilijk gespecificeerd worden met de bestaande methodologieën. Verder besteden de bestaande methodologieën weinig aandacht aan het schatten van geometrische parameters.

Bijdragen van dit proefschrift

In dit proefschrift wordt iTASC (*instantaneous Task Specification based on Constraints*) voorgesteld: een methodologie om sensorgebaseerde taken te specificeren. De voornaamste bijdragen van iTASC zijn:

- **de beperkingsgebaseerde aard:** iTASC is een echte beperkingsgebaseerde methodologie. Hierdoor is iTASC geschikt om taken te specificeren die verschillende bronnen van controlebeperkingen vereisen. In iTASC worden taken gespecificeerd volgens de *Task Function Approach* (Samson, Le Borgne, and Espiau 1991). Om deze manier van specificeren te ondersteunen worden de concepten van *objecten* en *kenmerken* ingevoerd, en controlebeperkingen gedefinieerd op de relatieve beweging van deze objecten en kenmerken. De methodologie laat exact, onder- en overgespecificeerde taken toe.

- **het generieke karakter:** iTASC is een generieke aanpak: de focus ligt niet op een bepaald soort taak, sensor of robot. iTASC kan algemeen gebruikt worden voor snelheidsgestuurde robots met rigide geledereren en gewrichten. Verder kunnen alle sensoren die geometrische informatie leveren ingepast worden in de methodologie (bijvoorbeeld camera's of afstandssensoren, maar ook dynamische sensoren zoals een krachtsensor, indien de taak quasistatisch verloopt).

- **de modelactualisatie:** Alle taakspecificatiemethodologieën gebruiken een geometrisch model van de taak, dat geactualiseerd moet worden tijdens het uitvoeren van de taak. Waar andere methodologieën erop rekenen dat hiertoe een ad-hocmethode geïmplementeerd wordt, voorziet iTASC een generieke actualisatieprocedure, om automatisch de pose van de object- en kenmerkassenstelsels te berekenen.

- **de integratie van schatting:** Daar waar overige methodologieën er doorgaans op rekenen dat het gebruik van metingen in de controle de taakuitvoering bestand maakt tegen geometrische variaties, biedt iTASC de primitieven aan om deze geometrische onzekerheden te modelleren en te schatten.

Om de effectiviteit van iTASC aan te tonen, wordt de methodologie toegepast op verschillende voorbeeldapplicaties.

2 Taakmodellering en -specificatie

Deze sectie beschrijft iTASC (*instantaneous Task Specification based on Constraints*): een methodologie om taken beperkingsgebaseerd te specificeren.

2.1 Controle gebaseerd op beperkingen

Samson, Le Borgne, and Espiau (1991) introduceerden de *Task Function Approach*, om robottaken te beschrijven. De idee achter deze aanpak is, dat robotcontrole neerkomt op het regelen van een –eventueel multidimensionale– functie, de *Task Function* genoemd. Een taak wordt gespecificeerd door deze Task Function $\boldsymbol{e}(\boldsymbol{q_R}, t)$, en de gewenste waarde $\boldsymbol{e}_{des}$ voor deze functie:

$$\boldsymbol{e}(\boldsymbol{q_R}, t) = \boldsymbol{e}_{des}. \tag{1}$$

Aangezien (1) een set (mogelijks niet-lineaire) beperkingen uitdrukt op de gewrichtsposities $\boldsymbol{q_R}$, wordt de benaming *beperkingsgebaseerde controle* gebruikt.

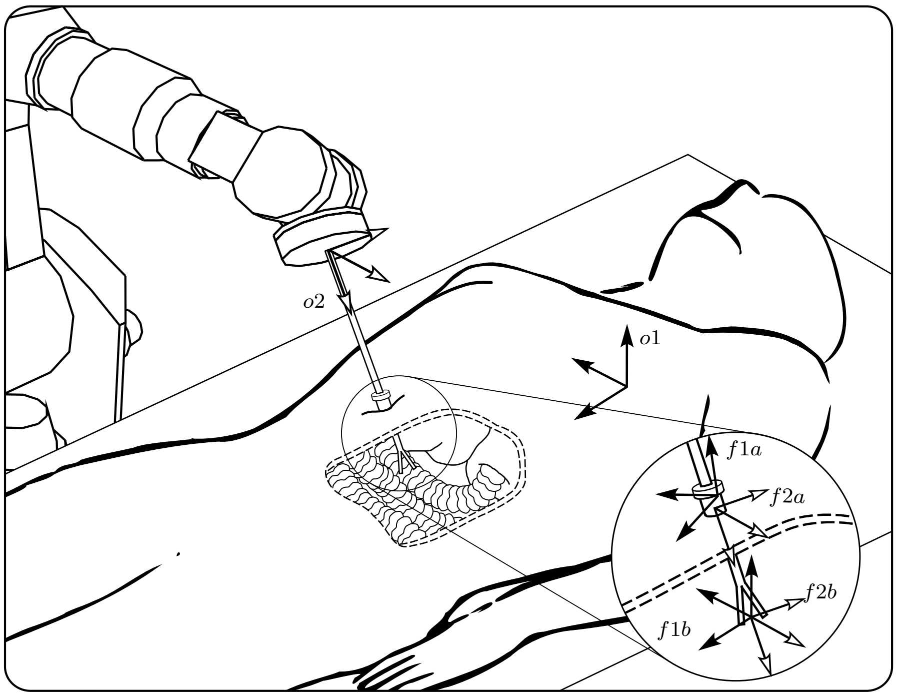
Voor een snelheidsgebaseerde robot genereert het controleprogramma de gewrichtssnelheden, die aan de robot aangelegd worden om de taak te realiseren. De eerste-orde afgeleide van (1) drukt de relatie –of: de *beperkingen*– uit, waaraan deze gewrichtssnelheden $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ moeten voldoen:

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q_R}} \dot{\boldsymbol{q}}_{\boldsymbol{R}} + \frac{\partial \boldsymbol{e}}{\partial t} = \dot{\boldsymbol{e}}_{des}. \tag{2}$$

Aangezien (2) lineair is in $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$, kunnen gekende optimalisatietechnieken aangewend worden, om $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ te berekenen (Doty, Melchiorri, and Bonivento 1993; Nakamura 1991).

Hoewel het concept op zich, om een taak te beschrijven in termen van een lineaire set beperkingen (2), eenvoudig is, is het niet noodzakelijk eenvoudig om de beperkingen zelf op te stellen. De meest generieke procedure is om (1) en de afgeleide (2) analytisch op te stellen. Deze methode is echter enkel praktisch bruikbaar voor eenvoudige taken. Om complexere taken (of dus, complexere beperkingen) te specificeren, is taakspecificatie-ondersteuning noodzakelijk. Deze sectie beschrijft iTASC: een methodologie om taken beperkingsgebaseerd te specificeren.

Om de specificatie van de beperkingen te ondersteunen, vertrekt de methodologie van een kinematisch model van de taak. Hiervoor worden de concepten



Figuur 1: De object- en kenmerkassenstelsels bij een toepassing voor een minimaal invasieve chirurgie.

van *objecten* en *kenmerken* ingevoerd. Hun relatieve beweging wordt gemodelleerd in termen van ogenblikkelijke basissen voor deze beweging, *kenmerk-jacobianen*, en de coördinaten in deze basissen, de *kenmerk-twistcoördinaten*[1]. Uiteindelijk worden de beperkingen gedefinieerd in termen van deze kenmerk-twistcoördinaten. De volgende sectie beschrijft iTASC aan de hand van een voorbeeldtoepassing.

2.2 De iTASC-methodologie

Deze sectie beschrijft de iTASC-methodologie aan de hand van een voorbeeldtoepassing. De methodologie bestaat uit vier stappen:

1. De keuze van de objecten en kenmerken.

[1] *Twist* is de Engelse benaming voor een 6D snelheidsvector $\mathbf{t}$, bestaande uit een translatie- en een rotatiesnelheid: $\mathbf{t} = [\boldsymbol{v}\ \boldsymbol{\omega}]^T$. Aangezien de correcte Nederlandse benaming, *kronkel*, zeer weinig gebruikt wordt, wordt hier de Engelse uitdrukking gebruikt.

2. De modellering van de relatieve beweging van de objecten en de kenmerken.

3. De definitie van de beperkingen.

4. Het oplossen naar de ogenblikkelijke beweging.

De voorbeeldtoepassing is een toepassing voor minimaal invasieve chirurgie (Figuur 1). Een robot met zes vrijheidsgraden manipuleert een laparoscopisch werktuig, dat een grijper heeft als uiteinde. Het werktuig wordt in het lichaam van een patiënt gebracht via een opening in de huid. Deze opening wordt de *trocar* genoemd. De beweging van de robot moet zo aangestuurd worden, dat het trocarpunt, of dus de intersectie van het werktuig en de patiënt, op de gewenste positie blijft. Verder moet de beweging van de robot zo gecontroleerd worden dat de grijper, of dus het uiteinde van het werktuig, langs een gespecificeerd traject beweegt om een orgaan in het lichaam van de patiënt te bereiken. Deze taakbeschrijving suggereert twee sets bewegingsbeperkingen: één gerelateerd aan de positie van het trocarpunt, en één gerelateerd aan de beweging van de grijper.

Stap 1: De keuze van de objecten en kenmerken

In het algemeen beschrijft de Task Function (1), of de eerste-orde uitdrukking van de Task Function (2), een aantal gelijktijdige subtaken waarin telkens *een zekere relatie* tussen *objecten* in de omgeving van de robot moet gecontroleerd worden, *door de robot te bewegen*. Een dergelijke relatie wordt een *taakrelatie* genoemd. In de voorbeeldtoepassing zijn er twee taakrelaties: (i) controleer de positie van het trocarpunt, en (ii) controleer de beweging van de grijper. In beide gevallen handelt het zich om bewegingen van het werktuig ten opzichte van het lichaam van de patiënt. Deze zijn dus de relevante objecten in de taak: het lichaam van de patiënt (object 1), en het werktuig (object 2), dat vast verbonden is aan de robot.

Na de keuze van de objecten worden de *kenmerken* van deze objecten gekozen. Een kenmerk is *dat onderdeel van het object, dat relevant is voor de taakrelatie.* Een kenmerk kan een fysiek onderdeel van het object zijn, zoals een hoekpunt, een zijde of een oppervlak, of een abstracte geometrische eigenschap van een fysiek onderdeel, zoals bijvoorbeeld de symmetrie-as van een cilinder of het referentie-assenstelsel van een sensor. In de voorbeeldtoepassing zijn er twee sets beperkingen en dus twee relevante kenmerken: het trocarpunt (kenmerk a) en het uiteinde van het werktuig (kenmerk b).

Stap 2: De modellering van de relatieve beweging van de objecten en de kenmerken

In de tweede stap worden de objecten en de kenmerken gemodelleerd door assenstelsels, en worden de relatieve bewegingen van deze assenstelsels beschreven in termen van basissen voor deze beweging, de *kenmerkjacobianen*, en coördinaten in deze basissen, de *kenmerk-twistcoördinaten*.

Voor elke taakrelatie worden twee objectassenstelsels ingevoerd, $o1$ en $o2$, en twee kenmerkassenstelsels, $f1$ en $f2$. De regels om deze assenstelsels te definiëren zijn:

1. $o1$ en $o2$ zijn vast verbonden met de objecten,
2. $f1$ en $f2$ zijn verbonden met de objecten, maar niet noodzakelijk *vast* verbonden,
3. de verbinding $o1 \rightarrow f1 \rightarrow f2 \rightarrow o2$ vormt een kinematische ketting; de zes vrijheidsgraden tussen $o1$ en $o2$ zijn verspreid over drie deelbewegingen: deelbeweging I van $f1$ ten opzichte van $o1$, deelbeweging II van $f2$ ten opzichte van $f1$ en deelbeweging III, van $o2$ ten opzichte van $f2$.

Verder worden de assenstelsels zo gekozen, dat eenvoudige wiskundige uitdrukkingen voor de deelbewegingen bekomen worden.

Figuur 1 toont de assenstelsels voor de voorbeeldtoepassing:

- Assenstelsel $o1$ geeft een referentiepositie op het lichaam van de patiënt weer,
- Assenstelsel $o2$ is bevestigd aan de robot. De oorsprong van $o2$ valt samen met het punt waar het laparoscopisch werktuig bevestigd is, en de Z-as van $o2$ is georiënteerd langs de as van het werktuig.
- Assenstelsel $f1a$ bevindt zich op een vaste positie ten opzichte van $o1$. De oorsprong van $f1a$ geeft de *gewenste* positie voor het trocarpunt weer, en de Z-as van $f1a$ staat loodrecht op het lichaam van de patiënt.
- Assenstelsel $f2a$ bevindt zich op het laparoscopisch werktuig. De oorsprong van $f2a$ valt samen met het *werkelijk* trocarpunt, en de oriëntatie van $f2a$ is dezelfde als die van $o2$.
- Beide assenstelsels $f1b$ en $f2b$ hebben hun oorsprong op het uiteinde van het werktuig. De oriëntatie van $f1b$ is dezelfde als die van $o1$, en de oriëntatie van $f2b$ is dezelfde als die van $o2$.

Voor elk van de kenmerken worden de deelbewegingen tussen $o1$ en $o2$ voorgesteld door een ogenblikkelijke twist: $\mathbf{t}_{o1}^{f1}, \mathbf{t}_{f1}^{f2}$ en $\mathbf{t}_{f2}^{o2}$. Voor elk van deze twists $i = I, II$ of III wordt een basis $\boldsymbol{J_{Fi}}$ gedefinieerd, die de bewegingsruimte van de deelbeweging omspant. De basissen $\boldsymbol{J_{Fi}}$ worden kenmerkjacobianen genoemd. De coördinaten van de deelbewegingen in deze basissen zijn de kenmerk-twistcoördinaten $\boldsymbol{\tau}_i$:

$$\mathbf{t}_{o1}^{f1} = \boldsymbol{J_{FI}}\boldsymbol{\tau}_I, \quad \mathbf{t}_{f1}^{f2} = \boldsymbol{J_{FII}}\boldsymbol{\tau}_{II}, \quad \mathbf{t}_{f2}^{o2} = \boldsymbol{J_{FIII}}\boldsymbol{\tau}_{III}. \tag{3}$$

Voor de voorbeeldtoepassing worden de kenmerkjacobianen als volgt gedefinieerd voor kenmerk a:

- Geen beweging tussen $f1a$ en $o2$:

$$\begin{aligned} \mathbf{t}_{o1}^{f1a} &= \boldsymbol{J}_{\boldsymbol{FI}}^{a}\boldsymbol{\tau}_I{}^{a}, \\ &= \mathbf{0}. \end{aligned} \tag{4}$$

- Twee vrijheidsgraden in translatie en drie in rotatie tussen $f2a$ en $f1a$:

$$\begin{aligned} {}^{f2a}_{f1a}\mathbf{t}_{f1a}^{f2a} &= \boldsymbol{J}_{\boldsymbol{FII}}^{a}\boldsymbol{\tau}_{II}{}^{a}, \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_1^a \\ \tau_2^a \\ \tau_3^a \\ \tau_4^a \\ \tau_5^a \end{bmatrix}. \end{aligned} \tag{5}$$

- Eén vrijheidsgraad in translatie tussen $f2a$ en $o2$:

$$\begin{aligned} {}^{f2a}_{o2}\mathbf{t}_{o2} &= \boldsymbol{J}_{\boldsymbol{FIII}}^{a}\boldsymbol{\tau}_{III}{}^{a}, \\ &= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \tau_6^a \end{bmatrix}. \end{aligned} \tag{6}$$

En voor feature b:

- Drie vrijheidsgraden in translatie tussen $f1b$ en $o1$:

$$\begin{aligned} {}^{f1b}_{o1}\mathbf{t}_{o1}^{f1b} &= \boldsymbol{J}_{\boldsymbol{FI}}^{b}\boldsymbol{\tau}_I{}^{b}, \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tau_1^b \\ \tau_2^b \\ \tau_3^b \end{bmatrix}, \end{aligned} \tag{7}$$

- Drie vrijheidsgraden in rotatie tussen $f2b$ en $f1b$:

$$\begin{aligned} {}_{f1b}^{f2b}\mathbf{t}_{f1b}^{f2b} &= \boldsymbol{J}_{\boldsymbol{F}II}^{b} \boldsymbol{\tau}_{II}{}^{b}, \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_4^b \\ \tau_5^b \\ \tau_6^b \end{bmatrix}, \end{aligned} \tag{8}$$

- Geen beweging tussen $f2b$ en $o2$:

$$\begin{aligned} \mathbf{t}_{o2}^{f2b} &= \boldsymbol{J}_{\boldsymbol{F}III}^{b} \boldsymbol{\tau}_{III}{}^{b}, \\ &= \mathbf{0}. \end{aligned} \tag{9}$$

Merk op dat de keuze van de object- en kenmerkassenstelsels en de keuze van het referentie-assenstelsel en referentiepunt van de twists leidt tot wiskundig eenvoudige uitdrukkingen voor de kenmerkjacobianen.

Aangezien $o2$ bevestigd is aan de robot, kan $\mathbf{t}_w^{o2}$ uitgedrukt worden in termen van de Jacobiaan van de robot $\boldsymbol{J_R}$ en de gewrichtssnelheden $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$:

$${}_{w}\mathbf{t}_w^{o2} = \boldsymbol{J_R}\dot{\boldsymbol{q}}_{\boldsymbol{R}}. \tag{10}$$

Hierin is w een inertieel referentie-assenstelsel.

De beweging van $o1$ hangt af van de bewegingen van de patiënt en wordt verondersteld gekend te zijn, bijvoorbeeld uit sensormetingen.

Stap 3: De definitie van de beperkingen

In de derde stap worden de beperkingen gedefinieerd die de gewenste bewegingen opleggen. In de voorbeeldtoepassing zijn twee sets van beperkingen noodzakelijk: één om de positie van de trocar te regelen, en één voor de beweging van het uiteinde van het werktuig:

- τ_1^a en τ_2^a drukken de X en Y-snelheid van $f2a$ ten opzichte van $f1a$ uit, of dus van het werkelijke trocarpunt ten opzichte van de gewenste positie. Om de gewenste positie van het trocarpunt te regelen worden de volgende beperkingen gedefinieerd:

$$\tau_1^a = k_{fb}(x_{desired}^a - x_{actual}^a), \tag{11}$$

$$\tau_2^a = k_{fb}(y_{desired}^a - y_{actual}^a). \tag{12}$$

  Hierin is k_{fb} een terugkoppelconstante, zijn x_{actual}^a en y_{actual}^a de coördinaten van de oorsprong van $f2a$, uitgedrukt in $f1a$, en zijn $x_{desired}^a$

en $y^a_{desired}$ de gewenste waardes voor deze coördinaten. Aangezien geen beweging van de trocar gewenst is, zijn $x^a_{desired}$ en $y^a_{desired}$ in deze toepassing constanten.

- Drie andere beperkingen zijn nodig om de translatiebeweging van het eindpunt van het werktuig op te leggen. Deze zijn analoog aan de vorige beperkingen, maar hebben betrekking tot τ^b_1, τ^b_2 en τ^b_3, aangezien deze coördinaten de X, Y en Z-snelheid van $f1b$ ten opzichte van $o1$ uitdrukken, of dus van het uiteinde van het werktuig ten opzichte van het lichaam van de patiënt:

$$\begin{aligned} \tau^b_1 &= k_{fb}(x^b_{desired} - x^b_{actual}), && (13)\\ \tau^b_2 &= k_{fb}(y^b_{desired} - y^b_{actual}), && (14)\\ \tau^b_3 &= k_{fb}(z^b_{desired} - z^b_{actual}). && (15) \end{aligned}$$

  Hierin zijn x^b_{actual}, y^b_{actual} en z^b_{actual} de x, y en z-coördinaten van $f1b$, uitgedrukt in $o1$. De gewenste waardes voor deze coördinaten, $x^b_{desired}$, $y^b_{desired}$ en $z^b_{desired}$, zijn tijdsafhankelijk. Ze worden gegenereerd door een padplanner, die een pad berekent tussen de initiële positie en de gewenste eindpositie.

Merk op dat de keuze van de object- en kenmerkassenstelsels, en van de kenmerkjacobianen, leidt tot eenvoudige uitdrukkingen voor de beperkingen.

Aangezien er in de voorbeeldtoepassing geen sensormetingen gebruikt worden, zijn er geen beperkingen die sensormetingen naar een gewenste waarde regelen. De definitie van dergelijke beperkingen vertrekt van de meetvergelijking van de sensor. De algemene beschrijving van de meetvergelijking voor geometrische sensoren wordt gegeven door:

$$\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{d}^{o2}_{o1}).$$

Deze meetvergelijking geldt ook voor dynamische sensoren, zoals een krachtsensor, indien de taak quasistatisch verloopt. De eerste-orde beschrijving van deze meetvergelijking wordt gegeven door:

$$\begin{aligned} \dot{\boldsymbol{z}} &= \frac{d\boldsymbol{g}}{d\boldsymbol{d}^{o2}_{o1}}\dot{\boldsymbol{d}}^{o2}_{o1}\\ &= \frac{d\boldsymbol{g}}{d\boldsymbol{d}^{o2}_{o1}}\boldsymbol{\mathcal{E}}\boldsymbol{J_F}\boldsymbol{\tau}\\ &\equiv \boldsymbol{J}_s\boldsymbol{\tau}. && (16) \end{aligned}$$

Vergelijking (16) definieert de *sensorjacobiaan* $\boldsymbol{J}_s$. De beperking om een sensorwaarde $\boldsymbol{z}$ te regelen naar een gewenste waarde $\boldsymbol{z}_{des}$ wordt dan gegeven door:

$$\boldsymbol{J}_s\boldsymbol{\tau} = \boldsymbol{u},$$

met $\boldsymbol{u}$ het resultaat van een controlewet, zoals, in het geval van proportionele controle, $\boldsymbol{u} = k_{fb}\left(\boldsymbol{z}_{des} - \boldsymbol{z}\right)$. De meest generieke manier om $\boldsymbol{J}_s$ te bekomen, is het analytisch afleiden van $\frac{d\boldsymbol{g}}{d\boldsymbol{\tau}}$. Voor veel praktische toepassingen kan $\boldsymbol{J}_s$ echter direct opgesteld worden.

Stap 4: Het oplossen van de ogenblikkelijke beweging

In de laatste stap wordt de set beperkingen (11)–(15) gecombineerd met de vergelijkingen van de relatieve beweging (4)–(10), zodat een set beperkingen (2) bekomen wordt. Deze set beperkingen wordt dan opgelost naar de gewrichtssnelheden $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$, die aangelegd worden aan de robot.

Definieer $\boldsymbol{J}^a_{\boldsymbol{F}}$ en $\boldsymbol{\tau}^a$ als:

$$\begin{aligned} \boldsymbol{J}^a_{\boldsymbol{F}} &= \left[\ {}_w\boldsymbol{J}^a_{\boldsymbol{F}I} \ \middle|\ {}_w\boldsymbol{J}^a_{\boldsymbol{F}II} \ \middle|\ {}_w\boldsymbol{J}^a_{\boldsymbol{F}III}\ \right], && (17)\\ \boldsymbol{\tau}^a &= \left[\ \tau^a_1 \quad \tau^a_2 \quad \ldots \quad \tau^a_6\ \right]^T. && (18) \end{aligned}$$

In (17) stelt ${}_w\boldsymbol{J}^a_{\boldsymbol{F}i}$ de kenmerkjacobiaan $\boldsymbol{J}^a_{\boldsymbol{F}i}$ voor, maar met w als referentieassenstelsel en de oorsprong van w als referentiepunt. $\boldsymbol{J}^b_{\boldsymbol{F}}$ en $\boldsymbol{\tau}^b$ worden analoog gedefinieerd.

Voor beide kenmerken geldt: $\mathbf{t}^{o2}_{o1} = \mathbf{t}^{f1}_{o1} + \mathbf{t}^{f2}_{f1} + \mathbf{t}^{o2}_{f2}$. Met (3), (17) en (18) leidt dit tot:

$$_w\mathbf{t}^{o2}_{o1} = \boldsymbol{J}^a_{\boldsymbol{F}}\boldsymbol{\tau}^a, \tag{19}$$

en tot een gelijkaardige uitdrukking voor kenmerk b:

$$_w\mathbf{t}^{o2}_{o1} = \boldsymbol{J}^b_{\boldsymbol{F}}\boldsymbol{\tau}^b. \tag{20}$$

Beschouw nu de gesloten kinematische ketting $w{\to}o1{\to}f1{\to}f2{\to}o2{\to}w$. De snelheidskringloopvergelijking voor deze ketting is:

$$\mathbf{t}^{o1}_w + \mathbf{t}^{f1}_{o1} + \mathbf{t}^{f2}_{f1} + \mathbf{t}^{o2}_{f2} + \mathbf{t}^{w}_{o2} = \mathbf{0},$$

of:

$$\mathbf{t}^{o1}_w + \mathbf{t}^{o2}_{o1} + \mathbf{t}^{w}_{o2} = \mathbf{0}.$$

Gecombineerd met (10), (19) en (20), leidt dit tot:

$$\begin{aligned} {}_w\mathbf{t}^{o1}_w + \boldsymbol{J}^a_{\boldsymbol{F}}\boldsymbol{\tau}^a - \boldsymbol{J}_{\boldsymbol{R}}\dot{\boldsymbol{q}}_{\boldsymbol{R}} &= \mathbf{0},\\ {}_w\mathbf{t}^{o1}_w + \boldsymbol{J}^b_{\boldsymbol{F}}\boldsymbol{\tau}^b - \boldsymbol{J}_{\boldsymbol{R}}\dot{\boldsymbol{q}}_{\boldsymbol{R}} &= \mathbf{0}. \end{aligned}$$

In matrixnotatie wordt dit geschreven als:

$$\begin{bmatrix} \boldsymbol{J}^a_{\boldsymbol{F}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{J}^b_{\boldsymbol{F}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\tau}^a \\ \boldsymbol{\tau}^b \end{bmatrix} - \begin{bmatrix} \boldsymbol{J}_{\boldsymbol{R}} \\ \boldsymbol{J}_{\boldsymbol{R}} \end{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{R}} = - \begin{bmatrix} {}_w\mathbf{t}^{o1}_w \\ {}_w\mathbf{t}^{o1}_w \end{bmatrix},$$

of, met invoering van de notaties $\bar{\boldsymbol{J}}_{\boldsymbol{F}}$, $\bar{\boldsymbol{\tau}}$, $\bar{\boldsymbol{J}}_{\boldsymbol{R}}$ en $\bar{\boldsymbol{T}}_u$:

$$\bar{\boldsymbol{J}}_{\boldsymbol{F}}\bar{\boldsymbol{\tau}} - \bar{\boldsymbol{J}}_{\boldsymbol{R}}\dot{\boldsymbol{q}}_{\boldsymbol{R}} = \bar{\boldsymbol{T}}_u. \tag{21}$$

Aangezien $\bar{\boldsymbol{J}}_{\boldsymbol{F}}$ steeds van volle rang is, want opgebouwd uit de matrices $\boldsymbol{J}_{\boldsymbol{F}}^a$ en $\boldsymbol{J}_{\boldsymbol{F}}^b$ die basissen zijn van volle rang, leidt (21) tot een uitdrukking voor $\bar{\boldsymbol{\tau}}$:

$$\bar{\boldsymbol{\tau}} = \bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\left(\bar{\boldsymbol{T}}_u + \bar{\boldsymbol{J}}_{\boldsymbol{R}}\dot{\boldsymbol{q}}_{\boldsymbol{R}}\right). \tag{22}$$

Met definitie van $\bar{\boldsymbol{u}}$ en $\bar{\boldsymbol{C}}_{\boldsymbol{F}}$ als:

$$\begin{aligned}
\bar{\boldsymbol{u}} &= \begin{bmatrix} k_{fb}(x^a_{desired} - x^a_{actual}) \\ k_{fb}(y^a_{desired} - y^a_{actual}) \\ k_{fb}(x^b_{desired} - x^b_{actual}) \\ k_{fb}(y^b_{desired} - y^b_{actual}) \\ k_{fb}(z^b_{desired} - z^b_{actual}) \end{bmatrix}, \\
\bar{\boldsymbol{C}}_{\boldsymbol{F}} &= \left[\begin{array}{cc|cc} \boldsymbol{I}_{2\times 2} & \boldsymbol{0}_{2\times 4} & \boldsymbol{0}_{2\times 3} & \boldsymbol{0}_{2\times 3} \\ \boldsymbol{0}_{3\times 2} & \boldsymbol{0}_{3\times 4} & \boldsymbol{I}_{3\times 3} & \boldsymbol{0}_{3\times 3} \end{array}\right],
\end{aligned}$$

kan de set beperkingen (11)–(15) herschreven worden als:

$$\bar{\boldsymbol{C}}_{\boldsymbol{F}}\bar{\boldsymbol{\tau}} = \bar{\boldsymbol{u}}. \tag{23}$$

In deze voorbeeldtoepassing is $\bar{\boldsymbol{C}}_{\boldsymbol{F}}$ een *selectiematrix*, die de coördinaten τ_i voor elke beperking selecteert uit de volledige coördinaatvector $\bar{\boldsymbol{\tau}}$. Door (23) te combineren met (22), wordt een stelsel lineaire vergelijkingen in $\dot{\boldsymbol{q}}_{\boldsymbol{R}}$ bekomen:

$$\bar{\boldsymbol{C}}_{\boldsymbol{F}}\bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\left(\bar{\boldsymbol{T}}_u + \bar{\boldsymbol{J}}_{\boldsymbol{R}}\dot{\boldsymbol{q}}_{\boldsymbol{R}}\right) = \bar{\boldsymbol{u}},$$

of:

$$\left(\bar{\boldsymbol{C}}_{\boldsymbol{F}}\bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\bar{\boldsymbol{J}}_{\boldsymbol{R}}\right)\dot{\boldsymbol{q}}_{\boldsymbol{R}} = \left(\bar{\boldsymbol{u}} - \bar{\boldsymbol{C}}_{\boldsymbol{F}}\bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\bar{\boldsymbol{T}}_u\right). \tag{24}$$

In elke tijdstap worden de gewrichtssnelheden berekend uit dit stelsel vergelijkingen (Doty, Melchiorri, and Bonivento 1993; Nakamura 1991), en aangelegd aan de robot. Aangezien in deze voorbeeldtoepassing slechts vijf beperkingen gedefinieerd zijn is het stelsel vergelijkingen onderbepaald. Indien de gewogen pseudo-inverse gebruikt wordt om (24) op te lossen, moeten gewichten in de gewrichtsruimte gedefinieerd worden. Een andere mogelijkheid is om extra beperkingen te definiëren in de nulruimte van $\bar{\boldsymbol{C}}_{\boldsymbol{F}}\bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1}\bar{\boldsymbol{J}}_{\boldsymbol{R}}$.

3 Modelactualisatie

Om de set beperkingen (2) uit te rekenen moet op elke tijdstap de pose van alle object- en kenmerkassenstelsels bekend zijn. Deze sectie beschrijft een generieke methode om deze modelactualisatie uit te voeren. De methode bestaat uit twee stappen: predictie en correctie.

Predictie

De kenmerk-twistcoördinaten kunnen in elke tijdstap berekend worden uit de ogenblikkelijke gewrichtssnelheden (vergelijking (22)):

$$\bar{\boldsymbol{\tau}} = \bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1} \left(\bar{\boldsymbol{T}}_u + \bar{\boldsymbol{J}}_{\boldsymbol{R}} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \right).$$

Voor elk kenmerkassenstelsel van kenmerk i wordt de overeenkomstige twist gevonden door de subvector $\boldsymbol{\tau}_{I}{}^{i}$, $\boldsymbol{\tau}_{II}{}^{i}$ or $\boldsymbol{\tau}_{III}{}^{i}$ uit $\bar{\boldsymbol{\tau}}$ te vermenigvuldigen met de overeenkomstige kenmerkjacobiaan $\boldsymbol{J}_{\boldsymbol{F}I}^{i}$, $\boldsymbol{J}_{\boldsymbol{F}II}^{i}$ or $\boldsymbol{J}_{\boldsymbol{F}III}^{i}$. Bijvoorbeeld, voor $\mathbf{t}_{o1}^{f1}$:

$$\mathbf{t}_{o1}^{f1} = \boldsymbol{J}_{\boldsymbol{F}I} \boldsymbol{\tau}_{I}.$$

Gelijkaardige uitdrukkingen gelden voor de overige twists. Een predictie voor de pose van de kenmerkassenstelsels op tijdstip $k+1$ wordt bekomen door de poses op tijdstip k te integreren met deze twists:

$$\widetilde{\boldsymbol{T}}_{o1\ k+1}^{f1} = \exp\left(\mathbf{t}_{o1}^{f1} T_s\right) \widehat{\boldsymbol{T}}_{o1\ k}^{f1}, \tag{25}$$

$$\widetilde{\boldsymbol{T}}_{f1\ k+1}^{f2} = \exp\left(\mathbf{t}_{f1}^{f2} T_s\right) \widehat{\boldsymbol{T}}_{f1\ k}^{f2}, \tag{26}$$

$$\widetilde{\boldsymbol{T}}_{f2\ k+1}^{o2} = \exp\left(\mathbf{t}_{f2}^{o2} T_s\right) \widehat{\boldsymbol{T}}_{f2\ k}^{o2}. \tag{27}$$

Hierin drukt $\tilde{}$ een predictie uit, en $\hat{}$ een schatting. T_s is de tijdstap.

Correctie

De iteratieve toepassing van (25) – (26) leidt tot divergentie door ophoping van integratiefouten. Daarom worden de voorspelde poses aangepast in een correctiestap. De correctie is gebaseerd op de bijkomende informatie in de posekringloopverglijkingen:

$$\boldsymbol{T}_{w}^{o1}(\boldsymbol{q}_{\boldsymbol{R}}, \boldsymbol{\chi}_{\boldsymbol{U}})\ \boldsymbol{T}_{o1}^{f1}\ \boldsymbol{T}_{f1}^{f2}\ \boldsymbol{T}_{f2}^{o2}\ \boldsymbol{T}_{o2}^{w}\,(\boldsymbol{q}_{\boldsymbol{R}}, \boldsymbol{\chi}_{\boldsymbol{U}}) = \boldsymbol{I}_{4\times4}. \tag{28}$$

Hierin is $\boldsymbol{q}_{\boldsymbol{R}}$ de vector van de gewrichtsposities van de robot, en $\boldsymbol{\chi}_{\boldsymbol{U}}$ een vector met eventuele ongecontroleerde vrijheidsgraden die van invloed zijn op de positie van de objecten (bijvoorbeeld de positie van een lopende band die extern aangestuurd wordt). In deze sectie wordt $\boldsymbol{\chi}_{\boldsymbol{U}}$ gekend verondersteld. Fouten in de predictie zorgen voor het niet-sluiten van de kringloopvergelijking:

$$\boldsymbol{T}_{w}^{o1}(\boldsymbol{q}_{\boldsymbol{R}}, \boldsymbol{\chi}_{\boldsymbol{U}})\ \widetilde{\boldsymbol{T}}_{o1}^{f1}\ \widetilde{\boldsymbol{T}}_{f1}^{f2}\ \widetilde{\boldsymbol{T}}_{f2}^{o2}\ \boldsymbol{T}_{o2}^{w}\,(\boldsymbol{q}_{\boldsymbol{R}}, \boldsymbol{\chi}_{\boldsymbol{U}}) = \boldsymbol{\Delta}. \tag{29}$$

In deze vergelijking stelt $\boldsymbol{\Delta}$ een homogene transformatiematrix voor, die verschilt van $\boldsymbol{I}_{4\times4}$. De linearisatie van (29) wordt gegeven door de snelheidskringloopvergelijking (maar dan met eindige verplaatsingen):

$$\bar{\boldsymbol{J}}_{\boldsymbol{F}} \bar{\boldsymbol{\tau}}_{\Delta} + \bar{\boldsymbol{J}}_{\boldsymbol{R}}\ \Delta \boldsymbol{q}_{\boldsymbol{R}} + \bar{\boldsymbol{T}}_{u\Delta} = \bar{\mathbf{t}}_{\Delta}. \tag{30}$$

Hierin stelt $\bar{\boldsymbol{\tau}}_\Delta$ de variaties voor op de kenmerk-twistcoördinaten, $\bar{\boldsymbol{T}}_{u\Delta}$ de eindige verplaatsingen die overeenkomen met variaties van de ongecontroleerde vrijheidsgraden $\boldsymbol{\chi_U}$, en $\bar{\mathbf{t}}_\Delta$ de eindige verplaatsingen die overeenkomen met de homogene transformaties $\boldsymbol{\Delta}$.

Aangezien er tijdens de correctiestap geen fysische beweging is van de robot of van de ongecontroleerde vrijheidsgraden, zijn $\Delta \boldsymbol{q_R}$ en $\bar{\boldsymbol{T}}_{u\Delta}$ nulmatrices, en wordt (30) gereduceerd tot:

$$\begin{aligned} \bar{\boldsymbol{J}}_{\boldsymbol{F}} \bar{\boldsymbol{\tau}}_\Delta &= \bar{\mathbf{t}}_\Delta, \\ \Leftrightarrow \bar{\boldsymbol{\tau}}_\Delta &= \bar{\boldsymbol{J}}_{\boldsymbol{F}}^{-1} \bar{\mathbf{t}}_\Delta. \end{aligned} \tag{31}$$

Voor elke kenmerk worden de kenmerkassenstelsels nu aangepast met de overeenkomstige subvector uit $\bar{\boldsymbol{\tau}}_\Delta$:

$$\widehat{\boldsymbol{T}}^{f1}_{o1\ k+1} = \exp\left(\boldsymbol{J_{FI}}\ \boldsymbol{\tau}_{I\Delta}\right) \widetilde{\boldsymbol{T}}^{f1}_{o1\ k+1}, \tag{32}$$

$$\widehat{\boldsymbol{T}}^{f2}_{f1\ k+1} = \exp\left(\boldsymbol{J_{FII}}\ \boldsymbol{\tau}_{II\Delta}\right) \widetilde{\boldsymbol{T}}^{f2}_{f1\ k+1}, \tag{33}$$

$$\widehat{\boldsymbol{T}}^{o2}_{f2\ k+1} = \exp\left(\boldsymbol{J_{FIII}}\ \boldsymbol{\tau}_{III\Delta}\right) \widetilde{\boldsymbol{T}}^{o2}_{f2\ k+1}. \tag{34}$$

4 Schatting

In de vorige sectie werden reeds coördinaten $\boldsymbol{\chi_U}$ ingevoerd, die ongecontroleerde vrijheidsgraden voorstellen. Ook ongekende parameters kunnen met deze coördinaten voorgesteld worden. Deze sectie beschrijft hoe dergelijke parameters geschat kunnen worden op basis van sensormetingen.

De pose van de objectassenstelsels wordt gegeven door:

$$\boldsymbol{T}^{o1}_{w} = \boldsymbol{T}^{o1}_{w}(\boldsymbol{q_R}, \boldsymbol{\chi_U}), \tag{35}$$

$$\boldsymbol{T}^{o2}_{w} = \boldsymbol{T}^{o2}_{w}(\boldsymbol{q_R}, \boldsymbol{\chi_U}). \tag{36}$$

Analoog aan de kenmerkjacobianen worden voor de coördinaten $\boldsymbol{\chi_U}$ ook basissen $\boldsymbol{J_U}$ gedefinieerd, die uitdrukken hoe de objectassenstelsels bewegen bij tijdsvariatie van de coördinaten $\boldsymbol{\chi_U}$:

$$\mathbf{t}^{o1}_{o2} = \boldsymbol{J_R} \dot{\boldsymbol{q}}_{\boldsymbol{R}} + \boldsymbol{J_U} \dot{\boldsymbol{\chi}}_{\boldsymbol{U}}. \tag{37}$$

Beschouw bijvoorbeeld een taak waarbij een robot een ton moet lokaliseren. Als de verticale positie van de ton gekend is, kan de positie van de ton beschreven worden door drie coördinaten: de posities x en y in het vlak, en de rotatie rond de verticale, ϕ. Of dus:

$$\boldsymbol{T}^{o2}_{o1} = \boldsymbol{T}^{w}_{o1}\ (\boldsymbol{q_R})\ \boldsymbol{T}^{o2}_{w}(\boldsymbol{\chi_U}),$$

met

$$\boldsymbol{\chi_U} = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix}.$$

De beweging van het objectassenstelsel bij variaties van de ongekende coördinaten wordt dan beschreven door:

$$\begin{aligned} {}_w\mathbf{t}_w^{o2} &= \boldsymbol{J_U}\dot{\boldsymbol{\chi}}_{\boldsymbol{U}} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix}. \end{aligned}$$

4.1 Integratie van schatting in de modelactualisatie

De predictie- en correctiestap van de modelactualisatie worden uitgebreid voor de schatting van de ongekende parameters.

Predictie

In het algemeen wordt het systeemmodel, dat de waardes van $\boldsymbol{\chi_U}$ op tijdstip $k+1$ relateert aan die op tijdstip k, uitgedrukt door:

$$\boldsymbol{\chi_U}_{k+1} = \boldsymbol{f}(\boldsymbol{\chi_U}_k). \tag{38}$$

Bijvoorbeeld, bij het schatten van de positie van de ton zijn de coördinaten $\boldsymbol{\chi_U}$ constant, of dus:

$$\boldsymbol{\chi_U}_{k+1} = \boldsymbol{\chi_U}_k.$$

In een ander opstelling, waar de ton aangevoerd wordt door een lopende band, is een model van constante snelheid beter geschikt:

$$\boldsymbol{\chi_U}_{k+1} = \boldsymbol{\chi_U}_k + \dot{\boldsymbol{\chi}}_{\boldsymbol{U}} T_s.$$

Volgens het systeemmodel wordt een voorspelling $\boldsymbol{\chi_U}$ berekend:

$$\widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}k+1} = \boldsymbol{f}(\widehat{\boldsymbol{\chi}}_{\boldsymbol{U}k}).$$

Een voorspelling van de kenmerkassenstelsels wordt bekomen zoals uitgelegd in Sectie 3. De predictie van de poses van $o1$ en $o2$ wordt bekomen door $\widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}k+1}$ in te vullen in (35) en (36).

Correctie

Fouten in de predictie zorgen terug voor het niet-sluiten van de kringloopvergelijking:

$$\widetilde{\boldsymbol{T}}_{w}^{o1}(\boldsymbol{q_R}, \widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}})\ \widetilde{\boldsymbol{T}}_{o1}^{f1}\ \widetilde{\boldsymbol{T}}_{f1}^{f2}\ \widetilde{\boldsymbol{T}}_{f2}^{o2}\ \widetilde{\boldsymbol{T}}_{o2}^{w}\ (\boldsymbol{q_R}, \widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}}) = \boldsymbol{\Delta}. \tag{39}$$

De linearisatie van (39) wordt gegeven door de snelheidskringloopvergelijking, met eindige verplaatsingen:

$$\bar{\boldsymbol{J}}_{\boldsymbol{F}}\bar{\boldsymbol{\tau}}_{\Delta} + \bar{\boldsymbol{J}}_{\boldsymbol{R}}\ \Delta\boldsymbol{q_R} + \bar{\boldsymbol{J}}_{\boldsymbol{U}}\Delta\boldsymbol{\chi}_{\boldsymbol{U}} = \bar{\mathbf{t}}_{\Delta}. \tag{40}$$

Of, aangezien de gewrichtsposities gekend zijn ($\Delta\boldsymbol{q_R} = \mathbf{0}$):

$$\begin{bmatrix} \boldsymbol{J_F} & \boldsymbol{J_U} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\tau}}_{\Delta} \\ \Delta\boldsymbol{\chi}_{\boldsymbol{U}} \end{bmatrix} = \mathbf{t}_{\Delta}. \tag{41}$$

De meetvergelijking voor geometrische sensoren wordt gegeven door:

$$z = g(\boldsymbol{d}_{o1}^{o2}). \tag{42}$$

De tijdsafgeleide van (42) wordt gegeven door:

$$\begin{aligned} \dot{z} &= \frac{dg}{d\boldsymbol{d}_{o1}^{o2}}\dot{\boldsymbol{d}}_{o1}^{o2} \\ &= \frac{dg}{d\boldsymbol{d}_{o1}^{o2}}\boldsymbol{E}^{-1}\mathbf{t}_{o1}^{o2}. \end{aligned} \tag{43}$$

Hierin kan $\mathbf{t}_{o1}^{o2}$ steeds uitgedrukt worden als een lineaire combinatie van $\dot{\boldsymbol{q}}_{\boldsymbol{R}}, \boldsymbol{\tau}$ en $\mathbf{t}_u$, of dus, voor alle kenmerken:

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \bar{\boldsymbol{H}}_{\boldsymbol{R}} & \bar{\boldsymbol{H}}_{\boldsymbol{F}} & \bar{\boldsymbol{H}}_{\boldsymbol{U}} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{R}} \\ \bar{\boldsymbol{\tau}} \\ \dot{\boldsymbol{\chi}}_{\boldsymbol{U}} \end{bmatrix}. \tag{44}$$

Volgens (42) wordt een voorspelling voor de meting berekend:

$$\widetilde{\boldsymbol{z}} = \boldsymbol{g}(\widetilde{\boldsymbol{d}}_{o1}^{o2}). \tag{45}$$

Voor beperkte variaties $\Delta\boldsymbol{z} = (\widetilde{\boldsymbol{z}} - \boldsymbol{z})$ geldt:

$$\Delta\boldsymbol{z} = \begin{bmatrix} \bar{\boldsymbol{H}}_{\boldsymbol{R}} & \bar{\boldsymbol{H}}_{\boldsymbol{F}} & \bar{\boldsymbol{H}}_{\boldsymbol{U}} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{q_R} \\ \bar{\boldsymbol{\tau}}_{\Delta} \\ \Delta\boldsymbol{\chi}_{\boldsymbol{U}} \end{bmatrix}. \tag{46}$$

Of, aangezien de gewrichtsposities gekend zijn:

$$\Delta\boldsymbol{z} = \begin{bmatrix} \bar{\boldsymbol{H}}_{\boldsymbol{F}} & \bar{\boldsymbol{H}}_{\boldsymbol{U}} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\tau}}_{\Delta} \\ \Delta\boldsymbol{\chi}_{\boldsymbol{U}} \end{bmatrix}. \tag{47}$$

Uit (41) en (47) worden $\bar{\boldsymbol{\tau}}_\Delta$ en $\Delta\boldsymbol{\chi}_{\boldsymbol{U}}$ berekend:

$$\begin{bmatrix} \mathbf{t}_\Delta \\ \Delta\boldsymbol{z} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J_F} & \boldsymbol{J_U} \\ \boldsymbol{H_F} & \boldsymbol{H_U} \end{bmatrix}^{\#}_{\boldsymbol{W}} \begin{bmatrix} \bar{\boldsymbol{\tau}}_\Delta \\ \Delta\boldsymbol{\chi}_{\boldsymbol{U}} \end{bmatrix}. \tag{48}$$

Hierin is $\boldsymbol{W}$ een wegingsmatrix die het relatieve belang van de positie-kringloopvergelijking en de metingen in de correctie weergeeft. De object- en kenmerkassenstelsels voor elk kenmerk worden dan aangepast volgens:

$$\begin{aligned} \widehat{\boldsymbol{T}}^{f1}_{o1\ k+1} &= \exp\left(\boldsymbol{J_{FI}}\tau_{\Delta\,I}\right)\widetilde{\boldsymbol{T}}^{f1}_{o1\ k+1}, && (49)\\ \widehat{\boldsymbol{T}}^{f2}_{f1\ k+1} &= \exp\left(\boldsymbol{J_{FII}}\tau_{\Delta\,II}\right)\widetilde{\boldsymbol{T}}^{f2}_{f1\ k+1}, && (50)\\ \widehat{\boldsymbol{T}}^{o2}_{f2\ k+1} &= \exp\left(\boldsymbol{J_{FIII}}\tau_{\Delta\,III}\right)\widetilde{\boldsymbol{T}}^{o2}_{f2\ k+1}, && (51) \end{aligned}$$

en

$$\begin{aligned} \widehat{\boldsymbol{T}}^{o1}_{w\ k+1} &= \boldsymbol{T}^{o1}_{w}(\boldsymbol{q_R}, \widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}} + \Delta\boldsymbol{\chi}_{\boldsymbol{U}}), && (52)\\ \widehat{\boldsymbol{T}}^{o2}_{w\ k+1} &= \boldsymbol{T}^{o2}_{w}(\boldsymbol{q_R}, \widetilde{\boldsymbol{\chi}}_{\boldsymbol{U}} + \Delta\boldsymbol{\chi}_{\boldsymbol{U}}). && (53) \end{aligned}$$

5 Toepassingen

Deze sectie geeft een overzicht van de toepassingen die in dit proefschrift besproken worden, als voorbeelden van de toepassing van iTASC.

Minimaal invasieve chirurgie met acht vrijheidsgraden

Deze toepassing (pagina 81) is een uitbreiding van de voorbeeldtoepassing voor minimaal invasieve chirurgie, waarbij het laparoscopisch werktuig twee extra vrijheidsgraden heeft in het lichaam van de patiënt. Dit laat toe de grijper in zes dimensies te positioneren ten opzichte van een orgaan in het lichaam.

De focus bij deze taak ligt op de aanpassing van een bestaande toepassing aan een nieuwe opstelling: om de uitgebreidere taak te specificeren zijn slechts beperkte veranderingen nodig aan de specificatie van de voorbeeldtoepassing.

Vormgeving

In deze taak (pagina 85) wordt een plaat geplaatst tussen twee halve bollen, waarvan één gemanipuleerd wordt door de robot. Door de bollen over elkaar te rollen en een contactkracht te regelen, wordt de plaat omgevormd tot een gekromd oppervlak.

Deze taak is voornamelijk een voorbeeld van een taak waarbij sensormetingen (kracht) gebruikt worden in de controle.

Inspectie

In deze taak (pagina 89) voert een hoog-redundante robot een inspectietaak uit in een pijpleiding. De robot gaat de pijpleiding binnen langs een flens en oriënteert een camera naar een naad om een visuele controle uit te voeren. De gewrichten van de robot moeten één voor één door de flens passeren. Hiervoor is telkens een set beperkingen actief. Overige beperkingen realiseren de relatieve oriëntatie tussen de camera (op het uiteinde van de robot) en de naad.

In deze taak zijn beperkingen gelijkaardig aan die van de toepassing voor minimaal invasieve chirurgie. De focus bij deze taak ligt echter op de overgang tussen de discrete fases waarin telkens andere beperkingen actief zijn.

Meervoudig puntcontact

Het meervoudig puntcontact (pagina 94) is een typevoorbeeld van een taak die niet (eenvoudig) te beschrijven is met methodologieën zoals het Task Frame Formalism. In deze toepassing moet een werktuig een naad volgen, die gevormd wordt door twee oppervlakken. De vorm van het werktuig is niet dezelfde als die van de naad, waardoor het contact tussen het werktuig en de naad bestaat uit twee puntcontacten. De beperkingen regelen de contactkrachten en de relatieve beweging van het werktuig ten opzichte van de naad.

Lasergraveren met twee lasers

In deze toepassing (pagina 100) bestaat de taak eruit simultaan een pad te traceren op een vlak en een cilindrische ton, met twee lasers die vast verbonden zijn met de robot. De lasers meten tevens de afstand tot het vlak en de ton. De exacte posities van het vlak en de ton zijn initieel ongekend.

Bij deze deze voorbeeldtoepassing wordt de invloed van de correctiestap bij de modelactualisatie verduidelijkt. Verder bevat deze toepassing schatting van geometrische parameters.

Mens-robot gedeelde controle

Deze toepassing (pagina 108) is een voorbeeld van een overgespecificeerde taak, voor mens-robot gedeelde controle. Een robot helpt een operator om een zwaar machine-onderdeel te dragen en te positioneren in een assemblage. De operator interageert met de robot door krachten uit te oefenen op het machine-onderdeel. Om deze krachten te meten is een krachtsensor gemonteerd op het uiteinde van de robot. Het is de taak van de operator om één kant van het onderdeel uit te lijnen met de assemblage. De andere kant van het onderdeel wordt uitgelijnd door de robot, op basis van visuele informatie van een camera. De controle over de taak wordt dus gedeeld door de operator en de robot.

De focus bij deze toepassing ligt op het gebruik van meerdere sensoren om een taak uit te voeren, en de gedeelde controle tussen de mens en de robot, waarbij het dynamisch gedrag bepaald wordt door de weging van de beperkingen.

6 Besluit

Sensorgebaseerde robotica komt reeds verschillende jaren veelvuldig aan bod in onderzoekstoepassingen. Dit resulteerde in de ontwikkeling van verschillende taakspecificatiemethodologieën, zoals het *Task Frame Formalism* of een aantal methodologieën gebaseerd op hybride kracht/positiecontrole gedefinieerd in termen van een ogenblikkelijke kracht- en snelheidsbasis. Deze methodologieën zijn echter voornamelijk toegespitst op krachtcontrole. Verder is het niet eenvoudig om met deze methodologieën complexere taken te specificeren, zoals taken waarbij verschillende sensoren gelijktijdig gedeeltelijke informatie over de taak leveren, of taken waarbij gelijktijdig verschillende controlebeperkingen actief zijn, bijvoorbeeld voor verschillende delen van de robot. Ook gaat bij de bestaande methodologieën zo goed als geen aandacht uit naar het schatten van geometrische parameters.

In dit proefschrift wordt iTASC (*instantaneous Task Specification based on Constraints*) voorgesteld: een methodologie om sensorgebaseerde taken te specificeren, met inbegrip van schatting van geometrische parameters.

Bijdragen van dit proefschrift

De voornaamste bijdragen van iTASC zijn:

- **de beperkingsgebaseerde aard:** iTASC is een beperkingsgebaseerde methodologie. Hierdoor is iTASC geschikt om taken te specificeren die verschillende bronnen van controlebeperkingen vereisen. In iTASC worden taken gespecificeerd volgens de *Task Function Approach* (Samson, Le Borgne, and Espiau 1991). Om deze manier van specificeren te ondersteunen worden de concepten van *objecten* en *kenmerken* ingevoerd, en controlebeperkingen gedefinieerd op de relatieve beweging van deze objecten en kenmerken. De methodologie laat exact, onder- en overgespecificeerde taken toe.

- **het generieke karakter:** iTASC is een generieke aanpak: de focus ligt niet op een bepaald soort taak, sensor of robot. iTASC kan algemeen gebruikt worden voor snelheidsgestuurde robots met rigide geledederen en gewrichten. Verder kunnen alle sensoren die geometrische informatie leveren ingepast worden in de methodologie (bijvoorbeeld camera's of afstandssensoren, maar ook dynamische sensoren zoals een krachtsensor, indien de taak quasistatisch verloopt).

- **de modelactualisatie:** Alle taakspecificatiemethodologieën gebruiken een geometrisch model van de taak, dat geactualiseerd moet worden tijdens het uitvoeren van de taak. Waar andere methodologieën erop rekenen dat hiertoe een ad-hocmethode geïmplementeerd wordt, voorziet iTASC een generieke actualisatieprocedure, om automatisch de pose van de object- en kenmerkassenstelsels te berekenen.
- **de integratie van schatting:** Daar waar overige methodologieën er doorgaans op rekenen dat het gebruik van metingen in de controle de taakuitvoering bestand maakt tegen geometrische variaties, biedt iTASC de primitieven aan om deze geometrische onzekerheden te modelleren en te schatten.

Om de effectiviteit van iTASC aan te tonen, wordt de methodologie toegepast op verschillende voorbeeldapplicaties.

Beperkingen en toekomstig onderzoek

Deze sectie bespreekt de beperkingen van iTASC en geeft suggesties voor toekomstig onderzoek.

Toepassingsgebieden Het ultieme doel van de sensorgebaseerde robotica is het ontwikkelen van intelligente robots, die hun taken volledig autonoom kunnen uitvoeren in ongestructureerde omgevingen. De stand der techniek is nog ver van deze doelstelling verwijderd. Deze sectie bespreekt de toepassingsgebieden van iTASC, voor wat betreft de gebruikersdoelgroep en de robotomgevingen waarvoor de methodologie geschikt is.

- **Gebruikersdoelgroep** iTASC is een methodologie voor ogenblikkelijke controle. De methodologie steunt op de geometrische modellering van de taak, en de doelgroep voor iTASC bestaat dan ook vooral uit *systeemintegratoren*, die toepassingen ontwikkelen voor eindgebruikers. Systeemintegratoren kunnen de principes van iTASC aanleren, en het nodige inzicht verwerven om taken te specificeren volgens de methodologie. Om een taak te specificeren volgens iTASC is effectief inzicht vereist, omdat er geen vaste regels zijn om de object- en kenmerkassenstelsels en de kenmerkjacobianen te kiezen. Een goede keuze is noodzakelijk, omdat het moeilijk is de beperkingen te specificeren bij slecht gekozen kenmerk-twistcoördinaten. Dit is een nadeel, maar geen unieke eigenschap van de iTASC-methodologie. Ook bij andere specificatiemethodologieën is dergelijk inzicht noodzakelijk en verhindert dit de veelvuldige toepassing van deze methodologieën niet. Bijvoorbeeld, ook bij het *Task Frame Formalism* is het moeilijk een taak te specificeren, bij een slecht gekozen *Task Frame*.

Toch is het wenselijk de gebruiker verder te ondersteunen bij het opbouwen van het model van de taak. Hiertoe wordt specificatiesoftware ontwikkeld in onze onderzoeksgroep. In deze software worden de vrijheidsgraden tussen de object- en kenmerkassenstelsels voorgesteld door een kinematische ketting. De kenmerkjacobianen worden dan automatisch gegenereerd, zodat de wiskundige voorstelling van de gebruiker geabstraheerd wordt.

Voor *eindgebruikers* van een bepaalde robottoepassing blijft de iTASC-methodologie doorgaans volledig verborgen. Eindgebruikers interageren met de taak via een taakspecifieke interface, die ontworpen is door de systeemintegrator. In het algemeen laat een dergelijke interface de eindgebruiker toe parameters aan te passen, zoals bijvoorbeeld terugkoppelconstantes of de actieve toestand. Het interne model, zoals bijvoorbeeld de kenmerkjacobianen, blijft voor deze gebruikers verborgen.

- **Robotomgevingen** Volgens iTASC wordt een taak gespecificeerd door beperkingen en schatters te definiëren op basis van een model van de taak. Om de verschillende mogelijke toestanden in de taak te voorzien en de nodige beperkingen en schatters te definiëren voor elke toestand, moet de omgeving in zekere mate gestructureerd zijn. Dit is bijvoorbeeld het geval in industriële omgevingen, waar de mogelijke objecten in de omgeving van de robot bekend zijn, maar variatie mogelijk is op hun exacte posities. iTASC richt zich voornamelijk op dergelijke *semi-gestructureerde* omgevingen.

  Wanneer robots ingezet worden in *volledig ongestructureerde* omgevingen, zoals een huiselijke omgeving, is het onmogelijk om de beperkingen en schatters te specificeren voor elke mogelijke toestand. In dergelijke omgevingen is een hoger cognitief niveau vereist, dat in staat is de sensormetingen te interpreteren en autonoom strategische beslissingen te nemen. Hoewel reeds jaren onderzoek verricht wordt naar dergelijke artificiële intelligentie, zijn we nog ver verwijderd van echt intelligente systemen. In onze onderzoeksgroep worden schattingsmethodes ontwikkeld om geometrische modellen op te bouwen van volledig ongekende omgevingen, op basis van contactmetingen (kracht en positie) (Slaets et al. 2007). Een mogelijk onderwerp voor verder onderzoek is de integratie van deze methodes in iTASC. Een geplande handeling tussen twee objecten kan dan uitgevoerd worden van zodra deze objecten geïdentificeerd zijn in de omgeving van de robot.

  Een andere mogelijkheid om robots in te zetten in volledig ongestructureerde omgevingen, is een operator en een robot samen te laten werken om de taak te realiseren. Bijvoorbeeld, bij gebruik van een teleoperatie-opstelling kan de operator de robot tot bij de relevante objecten brengen, waarna de robot zijn taak verder autonoom uitvoert.

In elk geval kan iTASC gebruikt worden als de onderliggende methodologie, wanneer de omgeving van de robot in voldoende mate gestructureerd is zodat een model van de taak kan opgebouwd worden, en beperkingen en schatters gespecificeerd kunnen worden binnen dit model.

Methodologie voor ogenblikkelijke controle iTASC is een methodologie voor ogenblikkelijke controle. De gebruike modellen zijn ogenblikkelijke kinematische modellen van de taak, en worden gebruikt om ogenblikkelijke gewrichtssnelheden te berekenen. Hierdoor blijft padplanning beperkt tot het genereren van trajecten, en zijn de gebruikte optimalisatiecriteria in iTASC locale criteria, die een ogenblikkelijke norm minimaliseren, bijvoorbeeld van de gewrichtssnelheden of van de fout op de beperkingen. De uitbreiding van iTASC om globale padplanning en globale optimalisatie te ondersteunen is een uitdagend onderwerp voor verder onderzoek.

Focus op snelheidsgestuurde robots Volgens de iTASC-methodologie wordt een taak zo gespecificeerd, dat een eerste-orde uitdrukking van de *Task Function* bekomen wordt. Voor versnellingsgebaseerde robots is een tweede-orde uitdrukking nodig. De voornaamste redenen waarom dit proefschrift de nadruk legt op snelheidsgebaseerde robots zijn (i) dat vele taken gespecificeerd kunnen worden op snelheidsniveau, (ii) dat veel van de robots die aanstuurbaar zijn vanop een controle-PC effectief snelheidsgestuurde systemen zijn, en (iii) dat het veel moeilijker is inzicht te krijgen in de kenmerkjacobianen en deze te specificeren op het versnellingsniveau. Principieel staat echter niets de formulering van iTASC op versnellingsniveau in de weg (De Laet and De Schutter 2007).

Ongelijkheidsbeperkingen iTASC is gebaseerd op de specificatie van gelijkheidsbeperkingen om een taak te specificeren. Voor bepaalde toepassingen zijn ongelijkheidsbeperkingen echter wenselijk. Een voorbeeld is het verhinderen van botsingen: de afstand tussen de robot en een hindernis is in principe onbelangrijk, zolang deze boven een minimale (veilige) waarde blijft. Een mogelijke methode om dergelijke ongelijkheidsbeperkingen in iTASC te integreren is het toevoegen van deze beperkingen als gelijkheidsbeperkingen, die enkel geactiveerd worden als de ongelijkheid dreigt overtreden te worden.

Niet-lineaire schattingstechnieken De schattingstechnieken beschreven in Sectie 5.3 zijn gebaseerd op de linearisatie van de meetvergelijkingen, zoals bijvoorbeeld nodig voor een Kalman Filter. Een gelineariseerde aanpak is echter niet steeds geschikt voor sterk niet-lineaire schattingsproblemen. In dergelijke gevallen kunnen met overige technieken, zoals *particle filters* betere resultaten behaald worden.