Evaluation of Calibration Method for Redundant Dual Arm Industrial Robots Using Internal Sensors

Gustaf Bergström
Björn Green

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6012
ISSN 0280-5316

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

Printed in Sweden by Tryckeriet i E-huset
Lund 2017

Abstract

To maintain a high position accuracy in robots is essential in today's industry. Because of different factors, as for example wear and tear, the robots may need to be recalibrated after a while to maintain their high precision. Today the calibration can be costly. The robot is often sent away to be calibrated and will therefore cause stoppages in production, and if the calibration is performed in-house, education of staff and purchases of expensive external measurement equipment are required. In this thesis, a calibration method for the dual arm, seven degrees of freedom (per arm) robot YuMi [*YuMi* 2016] from ABB is evaluated. The idea is to connect the two arms to each other, perform specified movements and use solely the internal sensors in the robot to do the calibration. The arms will also be docked in fixed positions in the robot's work space and while fixed, they will be moved into different configurations to obtain additional measurement data. A challenging part is that the robot has an extra joint in each arm compared to common six-axis industrial robots, which makes the robot inherently redundant. The redundancy makes it possible to reach a certain point with the robot tool using several different arm configurations, this makes it more difficult to estimate the robot's parameters.

In this thesis, several different types of calibrations were performed. Because the calibration methods have some drawbacks that need to be further investigated, the correct parameters were only identified in some cases. In those cases where the real parameters were not found, only the repeatability, and not the accuracy, was improved. The methods showed these results in simulations, while the actual robot was not successfully calibrated in experiments.

Acknowledgments

We would like to give special thanks to our main supervisors, Björn Olofsson and Olof Sörnmo at the Department of Automatic Control, LTH, Lund University and Andreas Stolt at Cognibotics AB for their expertise in robotics and for always taking the time to discuss problems and helping us when in need. We would also like to thank Martin Holmstrand and Sandra Collin at Cognibotics AB for their help with practical challenges during the course of the thesis. A special thanks to our examiner Professor Anders Robertsson at the Department of Automatic Control, LTH, Lund University who, even though he is a very busy man, always tried to take the time and help out with theoretical and practical problems that we encountered. We also like to thank Fredrik Bagge at the Department of Automatic Control, LTH, Lund University whose work has been a big inspiration in this thesis. And lastly we would like to thank Klas Nilsson, CEO at Cognibotics AB, who made this master thesis possible.

Contents

1

Introduction

1.1 Nomenclature

The nomenclature used in this thesis is presented in Table 1.1.

Table 1.1 Nomenclature used in this thesis.

Term	Explanation
Configuration	The entire robot (arm) position and orientation including joint angles.
DOF	Degrees of freedom.
POE	Product of exponentials.
TCP	Tool center point.
Pose	Position and orientation of the TCP.
$\mathbf{J}$	Identification Jacobian.
$\mathbf{J}_i$	Jacobian corresponding to the ith joint.
$\mathbf{J}_{st}$	Jacobian corresponding to ξ_{st}.
q_i	Joint variable of the ith joint.
$\mathbf{q}$	Vector with joint angles.
$\delta\mathbf{q}$	Joint offset.
$\mathbf{T}$	Forward kinematics map.
$\mathbf{T}_a$	Actual end-effector pose.
$\mathbf{T}_n$	Nominal end-effector pose .
$\mathbf{T}_{st}$	Initial transformation matrix.
$\delta\mathbf{T}\mathbf{T}^{-1}$	Pose error at tool frame expressed in base frame.
$\widehat{\xi}_i$	Twist corresponding to the ith joint.
ξ_i	Twist coordinates corresponding to the ith twist.
ξ_{st}	Twist corresponding to initial transformation matrix.
ξ	Vector of all joint twist coordinates.
$\delta\xi$	Twist coordinate error.

1.2 Background

Robots are becoming a more natural occurrence in our everyday life. Particularly in the industry, robots are replacing humans to do tedious, repetitive, and dangerous tasks in for instance assembly lines for everything from cars to small electronic components. The robot needs to be able to perform a specific movement several hundred thousand times for this purpose. It is important that the model in the control system corresponds well to the actual robot, otherwise the programmed position will differ from the actual one. Factors like manufacturing tolerance, wear and tear, transmission errors, set-up errors, and compliance will affect the robot's accuracy. When a robot is delivered, a calibration is typically performed to cope with those factors, but after some use, wear and tear will increase the difference between the model and the actual robot. This is where robot calibration comes in. Robot calibration is a way to improve the accuracy, which is crucial if the same robot is to be used for a longer period of time.

Today, calibration is often performed with expensive, large and somewhat complicated cameras. Furthermore the calibration can take a while, especially when the knowledge and equipment is not available in-house. The standard procedure is to send the robot to a service center, which can lead to long downtime in production. Cognibotics AB is a company specialized in development of methods for determination of robot properties and specifications. Cognibotics is currently developing methods to calibrate robots without the use of external sensors. The only sensors being used in the calibration are those existing in the robot, i.e., joint angle measurements from the motors. The task in this thesis is to increase the accuracy of the robot IRB 14000, also called YuMi [*YuMi* 2016], from ABB.

1.3 Problem formulation

There is a lot of literature on different calibration methods for robots in books and online. They, however, often treat calibration of non redundant (see Section 2.3) robot manipulators with up to six joints in a single arm and often use external measurements systems such as laser trackers and cameras. The goal with this thesis is to evaluate a calibration method based on the product of exponentials (POE, see Sections 2.6 and 2.7) formula on a dual arm redundant robot with seven degrees of freedom (DOF) for each arm. This method has been verified to work on non-redundant single arm robots with less than seven DOF, with known position and orientation (pose) of the tool center point (TCP). The dual arm robot is a product from ABB and is called YuMi. The calibration method will be evaluated on YuMi's predecessor, a prototype called Frida [Kock et al., 2011]. In this thesis, the joints are numbered 1–7 counting from the base of the arm to the TCP[1]. The problem

[1] This numbering convention differs from ABB's where joints 1–6 correspond to the ones of a standard serial manipulator and joint 7 to the extra added joint between joint 2 and 3.

formulation of this master thesis is to:

1. Build a kinematic model for Frida using POE.
2. Evaluate the model.
3. Implement an algorithm for joint offset identification of a clamped (connected to one or several fixed point/points) arm.
4. Implement an algorithm for kinematic parameter identification for two arms connected with a clamping device.
5. Implement an algorithm joint offset identification for two arms connected with a clamping device.
6. Evaluate the algorithms in simulations and experiments.

1.4 Disposition

In Chapter 1, an introduction and background to the problem, problem formulation and some prerequisites are presented. In Chapter 2 theory behind the robot kinematics, derivation of the calibration algorithms and kinematic models are reviewed. In Chapter 3 a description of the experiments and simulations are presented, and the results are presented in Chapter 4. In Chapter 5, the results are discussed, and conclusions from the results are presented. In Appendix A the kinematic model used in the thesis and some information about YuMi can be found.

1.5 Python

Python is a high-level, open-source programming language. The fact that it is open-source implies that Python has a lot of third-party packages for different kind of needs. The ones used during the course of this thesis are presented below.

NumPy

NumPy [*NumPy* 2016] is a package in Python. It contains arrays and matrices and foremost, mathematical functions to operate on arrays and matrices. The linear algebraic package in NumPy has been most useful during this thesis.

SymPy

SymPy [*SymPy* 2016] is an extension to Python that brings symbolic mathematics to Python. It is possible to define symbolic expressions, and perform various mathematical operations on those.

SciPy

SciPy [*SciPy* 2015] is an open-source software for mathematics, science, and engineering that can be used with Python.

1.6 RobotStudio

RobotStudio [*RobotStudio* 2016] is a program from ABB, which enables the user to do off-line programming of robot systems from ABB. It uses ABB's VirtualController [*VirtualController*], which is a software implementation with the same functionality as the controller in the real robot. It is therefore possible to simulate all movements of the robot, before a program is uploaded to the real process.

1.7 RAPID

RAPID [*RAPID robot programming language*] is a high-level programming language used to control ABB robots. It is developed by ABB and a text editor is integrated in RobotStudio. Programs can be written in RobotStudio, and then uploaded to the actual robot.

MultiMove

The purpose of MultiMove [ABB, 2014] is to control several robots with the same controller. It opens up for solutions where one robot can hold and move an item, while another robot can work on the object. It is useful in this thesis, since the relation between Frida's two TCPs can be set to be fixed. This means that one arm can be set to follow the other, making it possible to have a virtual bar between the two TCPs.

Soft Servo

When Soft Servo [ABB, 2010b] [ABB, 2010a] is activated on a joint, the joint acts as a mechanical spring, with the force proportional to the deviation from the programmed position. Soft Servo is activated through the command SoftAct in RAPID code. A value between 0 and 100 defines the softness, 0 being no softness and 100 being full softness.

MoveL and MoveJ

When programming a movement in RAPID, two different types of movement commands can be used. With MoveL, the path between two targets will be linear. With MoveJ/MoveAbsJ, the path between two targets is calculated to minimize joint movement. Motion commands with MoveL are easier to interpret because of its linear path in the room, while MoveJ leads to an arc-like motion, which will be

hard to predict and care needs to be taken if moving the robot in narrow passages. However, MoveL will be more restricted to use close to singularities of the robot.

1.8 MATLAB

MATLAB [*MATLAB* 2016] language is designed for computational mathematics and is used in the program MATLAB, which has several toolboxes for various scientific purposes, such as robot kinematics for example.

Simulink

Simulink [*Simulink* 2016] is a block-diagram environment for simulation that is compatible with MATLAB. Models of various control systems can via a graphical interface be programmed and modified.

1.9 ExtCtrl

An ABB robot controller consists of two computers, the Main Computer (MC) and the Axis Computer (AXC). The MC handles high-level control while the AXC handles low-level control. ExtCtrl [Blomdell et al., 2010] is a protocol developed at LTH that makes it possible to intercept data sent between the MC and AXC and add external control signals to the robot. Simulink-built models are used to generate the C code used in ExtCtrl. ExtCtrl was used to log the joint values of the robot arm(s) during the experiments.

1.10 Frida

Frida [Kock et al., 2011] is a predecessor, a prototype, to YuMi and its specifications are somewhat different from those for YuMi. Some link lengths and their placement of the links are different, as well as the transformation from the robot base to the respective arm's base. The Frida robot has no model available in RobotStudio, but the Department of Automatic Control has from previous projects a model implemented in Simulink which can be used with ExtCtrl. Translated to the conventional joint numbering from 1–7, beginning from the base, the joints are numbered 1,2,7,3,4,5,6. This definition is adopted since ABB had not implemented the possibility to have robots with seven joints in RobotStudio, so they had to add the seventh joint as an external axis. The robot can be seen in Figure 1.1.

A vast difference between Frida and other ABB robots is the number of degrees of freedom. ABB robots usually have at most six DOF, while Frida has seven DOF. The extra DOF enables respective arm to reach each pose with several configurations. Another difference is the relationship between the base and the arm base. The base is usually defined at the foot of the robot, while the arm base is defined near

the beginning of the first link. For other ABB robots, these coincide, but for Frida, the base is located at the foot of the robot between the arms, while the arms are mounted a distance from the table with a slight outwards and upwards rotation as can be seen in Figure 1.1.

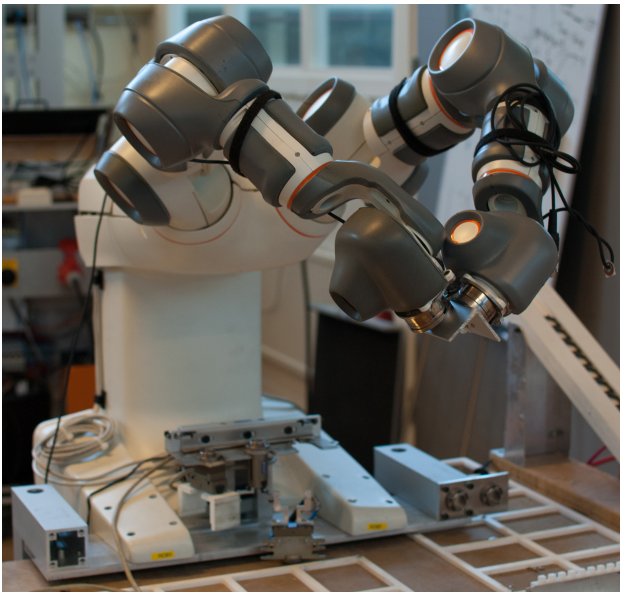

Figure 1.1 The robot Frida from ABB in the RobotLab of LTH, Lund University. The two arms are here rigidly connected (clamped together) via a connector plate with two tool changers.

2

Theory

2.1 Absolute accuracy and repeatability

In robotics, *absolute accuracy* and *repeatability* are main concepts. Absolute accuracy is the robot's ability to reach a specific point in the room. The accuracy is the deviation between the desired position and the actual position. The typical accuracy is somewhere around 8 mm to 15 mm [ABB, 2011] between the actual and the programmed positions. ABB have developed something called Absolute Accuracy [ABB, 2011], which is a way to decrease the discrepancy between the actual and the virtual robot by compensating the programmed position internally in the controller. This leads to an accuracy of about 0.5 mm in the entire working range. A robot can have good accuracy but poor repeatability, meaning that there is a large difference in pose between the times the target is trying to be reached, but the different poses are gathered around a certain target. A summary can be seen in Figure 2.1. The accuracy and repeatability are affected by factors like manufacturing tolerance, wear and tear, transmission errors, set-up errors, and joint and link compliance.

2.2 Different types of inaccuracy

Joint offset

Joint offset is the difference between the measured joint angle and the actual one. It can be a result from factory tolerances or set-up errors. It can also increase over time, because of wear.

Backlash

Each motor along the arm has a gear box. The cogs in the gearbox are not perfectly connected leading to a dead zone where a certain change of an input does not result in a well defined corresponding difference in the output but may change depending on if the cogs were in contact or not when starting the motion, see Figure 2.2. This is called backlash [*Lecture 8, Backlash and Quantization* 2015].

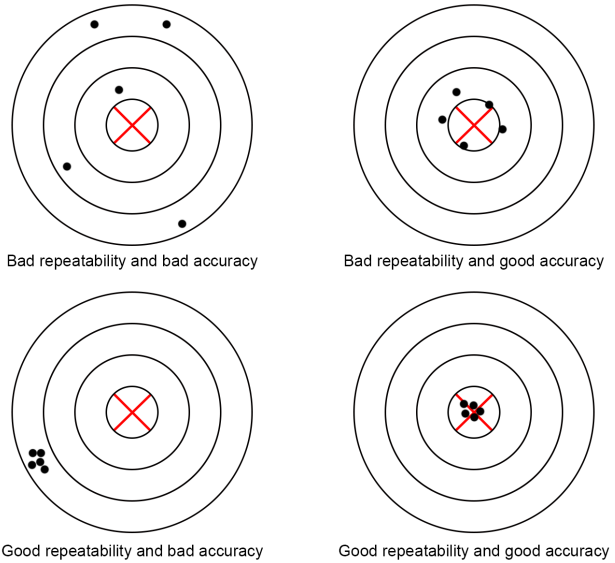

Figure 2.1 Accuracy and repeatability.

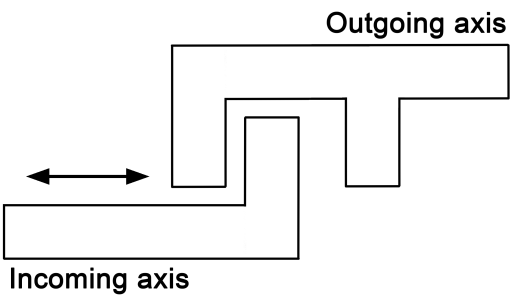

Figure 2.2 Schematic of backlash in a mechanical system.

Kinematic parameter errors

Kinematic parameter errors can be categorized in two subcategories. Firstly, the robot links can differ in length from the model of the robot. It can also be a result of factory tolerances or set-up errors. Secondly, the links' rotation axes can differ in orientation from the model. It can also be the result of factory tolerances or set-up errors. It can also be a result of damages, if the robot arm for example hits an obstacle with high speed.

Link flexing

Designing the robot links is a compromise between stiffness, weight, and cost. In a robot like YuMi, low weight is to be preferred because the motors are comparably weak. Because of the fact that the links are not perfectly stiff, they will flex a bit when exposed to force, as for example gravity from the links themselves or from a tool.

Joint flexing

The gearboxes in the joints imply flexibility, so just like the links, the joints are not totally stiff. A joint can therefore deflect from the given joint angle when the link connected to the joint is exposed to force.

2.3 Degrees of freedom (DOF)

For an open chain (attached to the ground at one single point) robot manipulator, the degree of freedom (DOF) is equal to the number of joints [Murray et al., 1994]. The typical robot from ABB has six DOFs, which enables it to have a dexterous workspace (the space where the robot can reach every position with arbitrary pose of the end-effector) [Spong et al., 2006]. For one less degree, there will be limitations on the poses that can be achieved.

Six DOFs is described in Figure 2.3, illustrating all possible movements in the 3D space that the body can perform. Three are for position and three are for orientation.

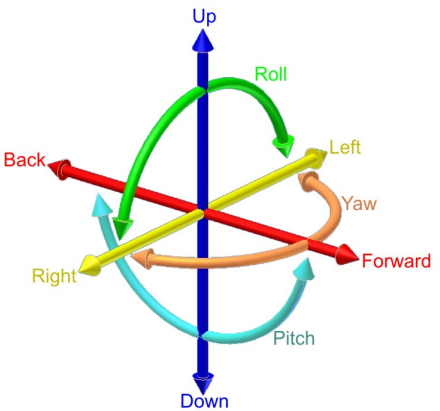

Figure 2.3 6 DOFs [Ionescu, 2010].

Redundancy

According to [Chiaverini et al., 2008], redundancy can be divided into two classes, namely:

Kinematic redundancy Kinematic redundancy arises when a robot manipulator has more DOFs than needed for a specific task. It can be useful, because it gives the opportunity to specify the configuration more freely to avoid obstacles, or to minimize the necessary joint movement. A six DOF robot can be kinematically redundant in certain tasks. An example is drilling, where the rotation around the drill axis does not matter.

Inherent redundancy Inherently redundant is a robot which has at least seven DOFs. Following a trajectory in space requires six DOFs, which means that a seven DOF robot can reach every pose on the trajectory with several different configurations. The Frida and YuMi robots both have seven DOFs on their respective arms, which make them both kinematically and inherently redundant. The purpose of kinematic redundancy is to make the robot more dexterous.

Redundant parameter

A redundant parameter model is a model where not all parameters can be identified individually. It might, for example, be possible to identify the sum of two parameters, but not the individual parameters. This fact means that the cost function for the problem can be minimized, but with ”wrong” parameters. The problem might converge to a local minimum, or even diverge if the redundancy has not been taken into consideration.

2.4 Base transformation

The base transformation is the transformation from the robot base (see Figure 2.5) to the arm base (see Figure 2.4). This transformation makes it possible to describe both arms' TCPs in the same coordinate frame.

Calculate the base transformation

The base transformation can be calculated from the relationship

$$\mathbf{T}_{base}\mathbf{T}_{arm}(\mathbf{q}) = \mathbf{T}_{point}(\mathbf{q}) \Longrightarrow \mathbf{T}_{base} = \mathbf{T}_{point}(\mathbf{q})\mathbf{T}_{arm}^{-1}(\mathbf{q})$$

where $\mathbf{T}_{base}$ is the desired transformation from the robot base to the arm base, $\mathbf{T}_{arm}$ is the transformation from the arm base to the TCP, $\mathbf{T}_{point}$ is the position of the TCP, read from the teach pendant (a control unit for the robot with touch screen and joystick), and $\mathbf{q}$ is the uncalibrated joint values.

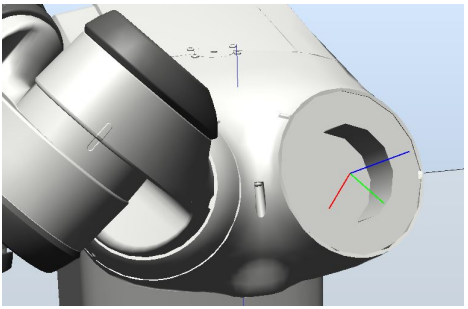

Figure 2.4 The arm base. Figure from [*RobotStudio* 2016]. The x-, y-, and z-axes are represented by the red, blue, and green axes, respectively.

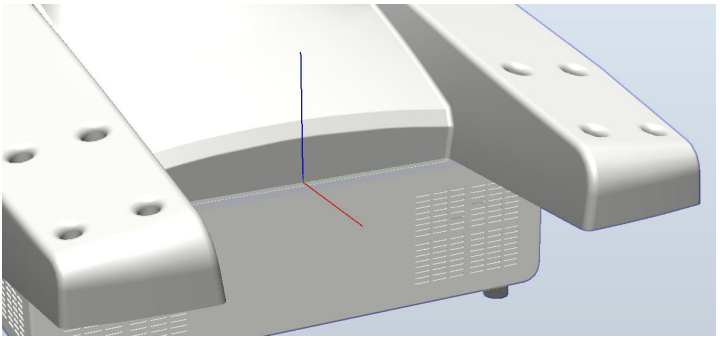

Figure 2.5 The robot base. Figure from [*RobotStudio* 2016]. The x-, y-, and z-axes are represented by the red, blue, and green axes, respectively.

2.5 Modeling and calibration

Robot calibration is, according to [Nof, 1999], divided into three levels as follows:

Level 1: Also called joint level calibration. In this calibration the relationship between the actual joint displacement and the recorded joint displacement.

Level 2: Kinematic calibration. The entire kinematic model of the robot is calibrated, including joint angle relationships and basic kinematic geometry of the robot.

Level 3: Non-kinematic calibration. This calibration compensates for errors due to effects such as joint compliance, friction and backlash.

In this thesis level 1 and level 2 calibration will be conducted. In level 2 calibration the parameters describing the robot's forward and inverse kinematics are identified why this also is called kinematic parameters. The steps for doing a kinematic calibration is:

1. Modeling of the robot kinematics.
2. Measurement of the end-effector pose (no explicit measurements done in this thesis).
3. Calibration of kinematic parameters.
4. Error compensation.

An essential part of parametric calibration is the establishment of an error model [Chen-Gang et al., 2014]. The error model is a way to describe the relationship between errors in the kinematic parameters and errors in the pose of the end-effector. A derivation of the error model for the POE model that is used in this thesis is performed in Section 2.9. The POE model is explained in more detail in Section 2.6

Robots can as almost anything be represented by mathematical models. There are several types of methods to model robots in the literature. Example of models [Chen-Gang et al., 2014] are the Denavit-Hartenberg model (DH), S model, CPC (Complete and parametrically continuous) model and POE model methods. There are three basic requirements that a kinematic model for robot calibration needs to meet [Chen-Gang et al., 2014]:

Completeness Meaning that the model needs to have enough parameters to describe any differences between the actual kinematic parameters and the nominal kinematic parameters.

Continuity A small change in the geometric properties of the manipulator should correspond to small changes in the kinematic parameters.

Minimality The model used for calibration should not include redundant parameters.

In the following subsections a short introduction to some of the different modeling methods is made.

Denavit-Hartenberg method

The DH representation for modeling robots has since its introduction been standard in the industry [Chen-Gang et al., 2014]. This is because kinematic models

established by this method is consistent despite who established them. The DH representation is centered around the assumptions that the coordinate frames assigned to two consecutive joints fulfill that [Spong et al., 2006][1]

1. The axis x_{n+1} (the x-coordinate for axis number $n+1$) is perpendicular to the axis z_n.
2. The axis x_{n+1} intersects the axis z_n.

The procedure of assigning frames to the links is not unique because of the fact that the origin and axes of each frame are arbitrary as long as it is rigidly attached to its corresponding link. Even though the parametrization can be done arbitrarily as long as it fulfills the above mentioned assumptions, the final result of the forward kinematics is always the same, hence the above mentioned consistency. By clever choices in assigning the frames, only four parameters per joint are needed to describe the homogeneous transformations between adjacent joints. The parameters are a_i, α_i, d_i, and θ_i and they describe link length, link twist, link offset, and joint angle, respectively. An arbitrary homogeneous transformation matrix uses six parameters, three for the translation and three for describing the rotation in difference to the DH representation. This fact and the systematic procedure in assigning reference frames are factors to why this is a popular method to use. The total number of parameters needed in the DH representation to describe the kinematics is given by $4r+2p+6$ [He et al., 2010], where r are the number of revolute joints, and p is the number of prismatic joints. This model contains no redundant parameters and hence fulfills the minimality condition. Because of constraints on the coordinate system when establishing the model, the four parameters are not enough to fulfill the completeness requirement stated above [Chen-Gang et al., 2014]. Another problem is that singularities arise when two consecutive joints are parallel, making the model not fulfill the continuity requirement. The interested reader can get a deeper knowledge on how the DH model works in [Spong et al., 2006].

S-model method

The S-model is an extension of the DH representation. Stone [Stone, 1987] added two parameters, which makes it a total of six parameters to describe the homogeneous transformations between adjacent joints and a total of $6n$ parameters to describe the entire kinematic chain of the robot. This fact makes the model complete, and the extra parameters also make it possible to place all coordinate systems on the robot (which is not always possible in the DH representation). However, the model is still not continuous and the extra parameters have made the calibration model redundant.

[1] Note that in some literature a so called *modified* DH-parametrization, with another index convention, is used.

Complete and parametrically continuous (CPC) method

The CPC method also adds two parameters to the DH model which makes the model complete and continuous, hence the name. The two extra parameters allow for an arbitrary positioning of the link coordinate frames, in contrast to the DH convention. All redundant parameters in this method can be systematically removed, making the method fulfill all the previously mentioned requirements for a kinematic model. There is, however, a rather user-unfriendly conditioning-handling technique needed to avoid singularities when constructing the error model. This is because the joint axes direction vectors are adopted as link parameters [Zhuang et al., 1993].

Product of exponentials

The POE formula was first introduced in robot calibration by Park and Okamura [Park and Okamura, 1994]. The POE formula is based on *Screw theory* and is explained in more detail in Sections 2.6 and 2.7. Park and Okamura's error model, however, was not on explicit form. An updated error model with explicit expressions for the generic error model was introduced by [He et al., 2010], which is used in this thesis. The POE formula fulfills all three requirements stated above.

2.6 Rigid body motion

An important part of robotics is rigid body motion [Murray et al., 1994]. Rigid motions describe a motion of a rigid object which preserves the distance between two points on that object. The motion in an Euclidean space is described by coordinates specified in three *orthonormal* (all axes perpendicular to one another and of unit length) axes, $(x,y,z) \in \mathbb{R}^3$. The distance between two points represented by coordinates $\mathbf{p_1} = (x_1,y_1,z_1)$ and $\mathbf{p_2} = (x_2,y_2,z_2)$ in Euclidean space is given by $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$. Preservation of a distance between two points, $\mathbf{p_1}$ and $\mathbf{p_2}$, in rigid motions then means that

$$\|\mathbf{p_2}(t)-\mathbf{p_1}(t)\| = \|\mathbf{p_2}(0)-\mathbf{p_1}(0)\| = \text{constant, for all } t$$

The movement of a rigid object obtained by a rigid motion is denoted a *rigid displacement*. A rigid body motion can consist of a translation and a rotation. Rotations in three dimensions can be represented in many different ways. One common way is Euler angles [Murray et al., 1994], but in this thesis, rotation matrices in the special orthogonal group $SO(3)$ will be used. The definition of $SO(3)$ is

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} : \mathbf{R}\mathbf{R}^T = \mathbf{I},\ \det\mathbf{R} = +1\} \tag{2.1}$$

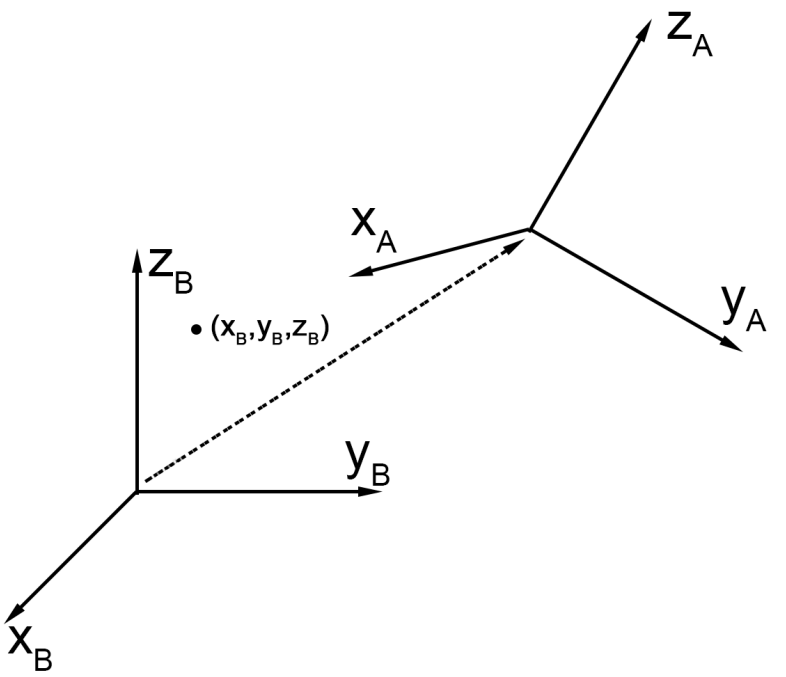

Figure 2.6 A translation and a rotation between two coordinate frames.

A rotation matrix can transform coordinates of a point (or frame axes) from one coordinate frame to another via matrix multiplication

$$\mathbf{p}_a = \mathbf{R}_{ab}\mathbf{p}_b = \begin{bmatrix}\mathbf{x}_{ab} & \mathbf{y}_{ab} & \mathbf{z}_{ab}\end{bmatrix}\begin{bmatrix}x_b \\ y_b \\ z_b\end{bmatrix} \tag{2.2}$$

where $\mathbf{x}_{ab}, \mathbf{y}_{ab}$, and $\mathbf{z}_{ab}$ are the coordinates of the principal axes of coordinate frame B and x_b, y_b, and z_b are the projected coordinates of the point onto reference frame B. In Figure 2.6 a translation and a rotation of a coordinate frame can be seen. A point on a rigid body with coordinate frame B attached to it can then be expressed in coordinate frame A by a rotation given by (2.2) and a displacement vector between the origins of the two frames. Hence, the rotation matrix $\mathbf{R}_{ab}$ together with the displacement vector rotate and move the coordinates of the point from frame B to frame A. This can be extended to include rotations and translations from several coordinate frames to one. Rotating between multiple coordinate frames is possible through the composition rule. A rotation matrix describing coordinates of frame C in frame B is given by $\mathbf{R}_{bc}$. Then the same coordinates can be represented in frame

A using the rotation matrix defined by the composition rule as:

$$\mathbf{R}_{ac} = \mathbf{R}_{ab}\mathbf{R}_{bc}. \tag{2.3}$$

A general rigid movement is a combined translation and rotation. Let $\mathbf{p}_{ab} \in \mathbb{R}^3$ be a displacement vector describing the position of frame B seen from frame A and $\mathbf{R}_{ab} \in SO(3)$ be the orientation of frame B relative to frame A. Then the general rigid movement is given by the pair $(\mathbf{p}_{ab}, \mathbf{R}_{ab})$ with a configuration space consisting of the product space of $\mathbb{R}^3$ and $SO(3)$ denoted as $SE(3)$. $SE(3)$ is the special Euclidean group defined by

$$SE(3) = \{(\mathbf{p}, \mathbf{R}) : \mathbf{p} \in \mathbb{R}^3, \mathbf{R} \in SO(3)\} = \mathbb{R}^3 \times SO(3) \tag{2.4}$$

The pair *(p,R)* describes a transformation of a point q, similar to that of (2.2), as

$$\mathbf{q}_a = \mathbf{p}_{ab} + \mathbf{R}_{ab}\mathbf{q}_b. \tag{2.5}$$

Homogeneous transformations

Transformations of points and vectors in $\mathbb{R}^4$ can be represented as homogeneous transformations. Homogeneous coordinates for points and vectors are given by

$$\mathbf{q}^H = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ 1 \end{bmatrix}, \quad \bar{v}H = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix}, \tag{2.6}$$

where the notation $\bar{v}$ is used to emphasize that it is a vector, not a point. The rigid movement described in (2.5) can be written on linear form, using above defined notations, as

$$\mathbf{q}_a^H = \begin{bmatrix} \mathbf{q}_a \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{ab} & \mathbf{p}_{ab} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{q}_b \\ 1 \end{bmatrix} \equiv \mathbf{T}_{ab}^H \mathbf{q}_b^H. \tag{2.7}$$

$\mathbf{T}_{ab}^H$, which is a 4×4 matrix (not to be confused with the Hermitian matrix), is called a *homogeneous representation* of a rigid movement. A composition rule exists for elements in $SE(3)$, similar as for the elements in $SO(3)$, and is given by

$$\mathbf{T}_{ac}^H = \mathbf{T}_{ab}^H \mathbf{T}_{bc}^H = \begin{bmatrix} \mathbf{R}_{ab}\mathbf{R}_{bc} & \mathbf{R}_{ab}\mathbf{p}_{bc} + \mathbf{p}_{ab} \\ 0 & 1 \end{bmatrix}. \tag{2.8}$$

Screws and twists

The screw theory is based on work by Chasles and Poinsot. Chasles proved that a rigid body motion, as described in the section about rigid body motion above, can be represented by a rotation around a straight axis and a translation parallel to the same axis. This is referred to as a *screw motion*. Poinsot's contribution to screw theory is the discovery that any system of forces acting on a rigid body can be substituted

by a single force applied along an axis together with a torque around the same axis. This force is known as a *wrench*. The POE formula is based on screw theory in combination with linear algebra and matrix groups and is based on Chasles' and Poinsot's theorems [Murray et al., 1994].

THEOREM 2.6.1
(Chasles' Theorem) [Murray et al., 1994] *Every rigid body motion can be realized by a rotation about an axis combined with a translation parallel to that axis.* □

THEOREM 2.6.2
(Poinsot's Theorem) [Murray et al., 1994] *Every collection of wrenches applied to a rigid body is equivalent to a force applied along a fixed axis plus a torque about the same axis.* □

Twists Twists ($\widehat{\xi}$) are infinitesimal screws and describe the instantaneous velocity of a rigid body by using its linear (v) and angular (ω) components. A rotation of a body around an axis can be derived by looking at the velocity of a point attached to a body rotating around an axis. The velocity is given by

$$\dot{\mathbf{q}}(t) = \omega \times \mathbf{q}(t) = \widehat{\omega}\mathbf{q}(t) \tag{2.9}$$

where ω is the rotation axis and the "hat"-operator, known as a "wedge", is just another way to represent the cross product of two vectors by turning the first factor into a matrix as

$$\mathbf{a} \times \mathbf{b} = \widehat{a}b = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix}. \tag{2.10}$$

The relation (2.9) can be integrated to give the point $\mathbf{q}$ at time t as

$$\mathbf{q}(t) = \exp(\widehat{\omega}t)\mathbf{q}(0), \tag{2.11}$$

where $\mathbf{q}(0)$ is the starting point. The matrix $\widehat{\omega}$ is a skew-symmetric matrix, meaning that it satisfies $\widehat{\omega}^T = -\widehat{\omega}$. The vector space spanned by all skew symmetric matrices is denoted $so(3)$ and is defined as

$$so(3) = \{\mathbf{S} \in \mathbb{R}^{3\times 3} : \mathbf{S}^T = -\mathbf{S}\}.$$

Using the series expansion of the exponential and the fact that ω is skew symmetric, the exponential is given by

$$\exp(\widehat{\omega}\theta) = \mathbf{I} + \widehat{\omega}\sin\theta + \widehat{\omega}^2(1 - \cos\theta) \tag{2.12}$$

provided that $\|\omega\| = 1$ and that $\theta \in \mathbb{R}$, which is the angle of rotation about ω. This is known as Rodrigues' formula [Murray et al., 1994]. Exponentials of elements

in $so(3)$ are orthogonal, so if $\mathbf{R} = \exp(\omega\theta)$ then $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $\det\mathbf{R} = +1$, i.e., if $\omega \in so(3)$ and $\theta \in \mathbb{R}$, then $e^{\widehat{\omega}\theta} \in SO(3)$. Another feature of the exponential map is that it is surjective onto $SO(3)$, which means that, given a rotation matrix $\mathbf{R}$, there exists an $\omega \in \mathbb{R}^3$, $\|\omega\| = 1$, and $\theta \in \mathbb{R}$ such that $\mathbf{R} = \exp(\widehat{\omega}\theta)$.

The exponential mapping can be generalized for the special Euclidean group $SE(3)$. From Figure 2.7, with $\omega \in \mathbb{R}^3$, $\|\omega\| = 1$, and $\mathbf{q}$ a point on the rotation axis ω, the velocity of point $\mathbf{p}$ can be written as

$$\dot{\mathbf{p}}(t) = \omega \times (\mathbf{p}(t) - \mathbf{q}(t)). \tag{2.13}$$

Converted to homogeneous coordinates the right-hand side of (2.13) is given by the 4×4 matrix

$$\widehat{\xi} = \begin{bmatrix} \widehat{\omega} & \mathbf{v} \\ 0 & 0 \end{bmatrix} \tag{2.14}$$

where $\mathbf{v} = -\omega \times \mathbf{q}$. (2.13) is then given by

$$\begin{bmatrix} \dot{\mathbf{p}} \\ 0 \end{bmatrix} = \begin{bmatrix} \widehat{\omega} & \mathbf{v} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \widehat{\xi} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \Rightarrow \dot{\bar{\mathbf{p}}} = \widehat{\xi}\bar{\mathbf{p}} \tag{2.15}$$

with the solution

$$\bar{\mathbf{p}}(t) = e^{\widehat{\xi}t}\bar{\mathbf{p}}(0) \tag{2.16}$$

where $\widehat{\xi}$ is a generalization of skew symmetric matrices $\widehat{\omega} \in so(3)$ and belongs to the group

$$se(3) = \{(v, \widehat{\omega} : \mathbf{v} \in \mathbb{R}^3, \widehat{\omega} \in so(3)\}. \tag{2.17}$$

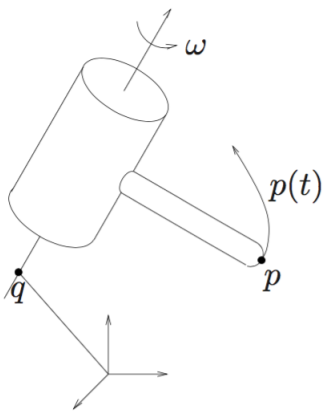

Figure 2.7 Revolute joint [Murray et al., 1994].

Elements belonging to $se(3)$ are referred to as *twists*. The twist coordinates of the twist can be retrieved from the homogeneous representation using the $\vee$ (vee)-operator

$$\begin{bmatrix} \widehat{\omega} & \mathbf{v} \\ 0 & 0 \end{bmatrix}^{\vee} = \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} = \xi \in \mathbb{R}^6 \tag{2.18}$$

while the $\wedge$(wedge)-operator works in the opposite direction.

$$\xi^{\wedge} = \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix}^{\wedge} = \begin{bmatrix} \widehat{\omega} & \mathbf{v} \\ 0 & 0 \end{bmatrix} = \widehat{\xi} \in se(3) \tag{2.19}$$

Similar to the mapping of elements in $so(3)$ to $SO(3)$, elements in $se(3)$ are mapped to $SE(3)$ via the exponential of $\widehat{\xi}\theta$ if $\widehat{\xi} \in se(3)$ and $\theta \in \mathbb{R}$,

$$e^{\widehat{\xi}\theta} \in SE(3). \tag{2.20}$$

An exponential mapping $e^{\widehat{\xi}\theta}$, with $\widehat{\xi} \in se(3)$ and $\theta \in \mathbb{R}$ can be written as

$$\mathbf{T} = e^{\widehat{\xi}\theta} = \begin{bmatrix} e^{\widehat{\omega}\theta} & (\mathbf{I} - e^{\widehat{\omega}\theta})(\omega \times \mathbf{v}) + \omega\omega^T \mathbf{v}\theta \\ 0 & 1 \end{bmatrix}, \quad \omega \neq 0. \tag{2.21}$$

Transformations of an element in $SE(3)$ is not, as for transformations between two rotational matrices in $SO(3)$, a mapping from one coordinate frame to another. Instead, it describes the mapping from the initial point or pose to the point or pose of the rigid body after the rigid motion is executed in the same coordinate frame:

$$\mathbf{p}(\theta) = e^{\widehat{\xi}\theta}\mathbf{p}(0) \tag{2.22}$$

Screws Screw motions are, as previously mentioned, a motion that consists of a rotation around and axis and a translation parallel to the same axis, see Figure 2.8. The components of the screw are an *axis of rotation* l defined by (2.25), a *pitch* h, (2.24) and a *magnitude* M, (2.26). The motion associated with the screw is given by the rotation $\theta = M$ (in the case of a revolute joint). In prismatic joints, θ is the amount of translation along the axis about the axis l and the translation $d = h\theta$ parallel to the axis. θ is the angle of rotation about the axis l. For a rigid body transformation of a point $\mathbf{p} \in \mathbb{R}^3$ using screw representation, the end location of the point is equal to

$$\mathbf{Tp} = \mathbf{q} + \exp(\widehat{\omega}\theta)(\mathbf{p} - \mathbf{q}) + h\theta\omega \tag{2.23}$$

where $\omega \in \mathbb{R}^3$ is the axis of the twist coordinates associated with the screw.

A twist $\xi = (\mathbf{v}, \omega) \in \mathbb{R}^6$ gives rise to the associated screw with screw coordinates (*pitch, h, axis, l and magnitude, M*) of the twist given by

$$h = \frac{\omega^T \mathbf{v}}{||\omega||^2} \tag{2.24}$$

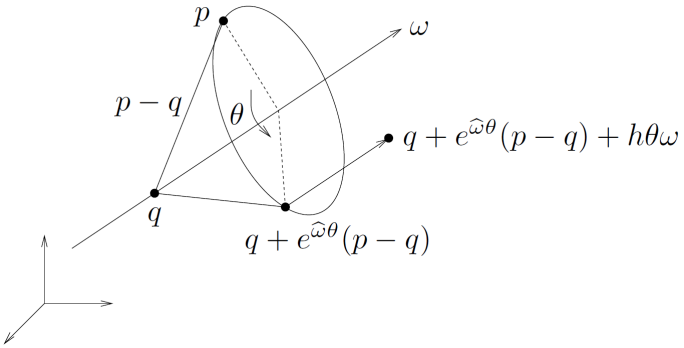

Figure 2.8 Generalized screw motion [Murray et al., 1994].

$$l = \begin{cases} \frac{\omega \times v}{||\omega||^2} + \lambda\omega : \lambda \in \mathbb{R} & \text{if } \omega \neq 0 \\ 0 + \lambda \mathbf{v} : \lambda \in \mathbb{R} & \text{if } \omega = 0 \end{cases} \tag{2.25}$$

where λ is a real number making (2.25) a directed line through the point $\frac{\omega \times \mathbf{v}}{||\omega||^2}$ with direction ω. Finally

$$M = \begin{cases} \|\omega\| & \text{if } \omega \neq 0 \\ \|v\| & \text{if } \omega = 0 \end{cases}. \tag{2.26}$$

The transformation in (2.23) given in homogeneous coordinates is

$$\mathbf{T} = \begin{bmatrix} e^{\widehat{\omega}\theta} & (\mathbf{I} - e^{\widehat{\omega}\theta})\mathbf{q} + h\theta\omega \\ 0 & 1 \end{bmatrix} \tag{2.27}$$

Exchanging $\mathbf{v} = -\omega \times \mathbf{q} + h\omega$ in (2.21) we get the same transformation matrix as in (2.27) and it is shown that the twist coordinates ξ generate the screw motion.

To represent a rigid transformation $\mathbf{T}$ dependent on a twist ξ in another coordinate frame, one can apply the adjoint transformation of the rigid transformation $\mathbf{T}$. The adjoint of $\mathbf{T}$ is given by

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ 0 & 1 \end{bmatrix} \rightarrow Ad_T = \begin{bmatrix} \mathbf{R} & \widehat{\mathbf{p}}\mathbf{R} \\ 0 & \mathbf{R} \end{bmatrix}. \tag{2.28}$$

It can be shown that the inverse of the adjoint is given by

$$Ad_T^{-1} = \begin{bmatrix} \mathbf{R}^T & -(\mathbf{R}^T\mathbf{p})^{\wedge}\mathbf{R}^T \\ 0 & \mathbf{R}^T \end{bmatrix} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\widehat{\mathbf{p}} \\ 0 & \mathbf{R}^T \end{bmatrix} = Ad_{T^{-1}} \tag{2.29}$$

This fact leads to a lemma

LEMMA 2.6.3—[MURRAY ET AL., 1994]
If $\widehat{\xi} \in se(3)$ *is a twist with twist coordinates* $\xi \in \mathbb{R}^6$, *then for any* $\mathbf{T} \in SE(3)$, $\mathbf{T}\widehat{\xi}\mathbf{T}^{-1}$ *is a twist with twist coordinates* $Ad_T\xi \in \mathbb{R}^6$. □

which is utilized in the derivation of the error model.

2.7 Product of exponentials representation

The transformation from base to end-effector of a robot can be described in many ways. As previously mentioned, DH-parameters are widely used. However, in this thesis product of exponentials will be used.

As mentioned in Section 2.5, the product of exponentials formula is based on screw theory. By representing the joints in the robot manipulator as twists, it is possible to express the forward kinematics of the robot using the theory of screws and twists in Section 2.6. The main advantages of using this representation is that it does not suffer from singularities and that it provides an intuitive geometric description of rigid body motion [Murray et al., 1994].

In the POE formulation there are only two coordinate frames, opposed to, for instance, the DH representation. There is one coordinate frame at the base of the robot and one at the tool. This is a due to that rigid transformations given by screw motions represent the rigid body transformations relative to the initial configuration. Given a generic n-DOF manipulator with n joints, the forward kinematics is given as [Okamura and Park, 1996]:

$$\mathbf{T} = \exp(\widehat{\xi}_1 q_1)\exp(\widehat{\xi}_2 q_2)\ldots\exp(\widehat{\xi}_n q_n)\mathbf{T}_{0,n}, \tag{2.31}$$

where $\widehat{\xi}$ comes from (2.14) and q_i is the rotation about the axis corresponding to twist i. As mentioned in Section 2.5 the POE formulation fulfills the three requirements for a kinematic model, i.e., completeness, continuity, and minimality.

2.8 Kinematic model

As mentioned in Section 2.6, the forward kinematics for a robot can be represented by twists associated with each joint, and screws are a geometric description of twists [Murray et al., 1994]. The relationship between twists and screws was discussed in Section 2.6. It is easier to assign screws representing the joints than twists. A screw is defined by its axis of rotation (ω) and a point $(\mathbf{q})$ on that axis. They can then be easily transformed into twist coordinates by setting the linear velocity component of the twist to $-\omega \times \mathbf{q}$.

By using a blueprint of the robot, one can use the lengths and orientations to assign screws to each joint. The representation used in this thesis can be seen in Section A.1.

2.9 Generic algorithm

All identification algorithms in this thesis are dependent on an initial guess. Because of the fact that the problem is non-linear, an initial guess too far from the correct value might result in the algorithm converging to a local minimum, and not the desired global minimum. Another problem with an initial guess too far away is that the solution might not converge at all. It is therefore crucial to use an initial guess reasonably close to the actual value. When identifying the kinematic parameters, the nominal values are used as an initial guess. For motor offsets, the initial guess used is zero.

Linear least-squares problems

Linear least squares problems are associated with solving the matrix equation [*Linear Least Squares Problems* 2013]

$$\mathbf{Ax} = \mathbf{b} \tag{2.32}$$

where $\mathbf{A}$ is an m-by-n matrix and $\mathbf{x}$ is an n-by-1 vector and $\mathbf{b}$ is an m-by-1 vector. This is done by finding the $\mathbf{x}$-vector minimizing the Euclidean norm (2-norm)

$$\|\mathbf{Ax} - \mathbf{b}\|_2. \tag{2.33}$$

Problems with $m > n$ gives that there are more equations than there are unknowns. These problems are called over-determined and there is normally no exact solution to these problems.

Least squares is used to approximate a solution to the over-determined system of equations by minimizing the sum of the squared errors in every equation in the system. These problems can be solved in some different ways, where two of the standard ways include *QR* decomposition and *Singular value decomposition*.

QR decomposition A QR decomposition is a decomposition of a matrix $\mathbf{A}$ into $\mathbf{A} = \mathbf{QR}$, where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ is an upper triangular matrix. QR decompositions are used to solve linear least squares problems [*Linear Least Squares Problems* 2013]. This can be utilized to check for linear dependence in matrices. If one of the diagonal elements in $\mathbf{R}$ is zero, then there is a linear dependency in the $\mathbf{A}$ matrix. The QR decomposition was used to analyze the linear dependence in the Jacobians during this thesis.

Singular value decomposition Singular value decomposition (SVD) [MIT, 2011] is a method where a matrix $\mathbf{M}$ is factorized to $\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^*$ where $\mathbf{U}$ is a unitary matrix, Σ is rectangular diagonal matrix with non-negative real numbers on the diagonal and $\mathbf{V}^*$ is a conjugate transpose of an unitary matrix. SVD is used in the built-in least squares solver from NumPy (see Section 1.5) that is utilized in the implementation to compute the parameter errors.

BFGS-method with a numerical Jacobian

SciPy's *optimize.minimize-function* [*SciPy* 2015] with the BFGS-method and a numerical Jacobian was used in this thesis. BFGS stands for Broyden, Fletcher, Goldfarb, and Shanno. This is one of the most powerful methods for solving non-linear optimization problems [*BFGS algorithm* 2013]. BFGS is an iterative quasi-Newton method which has proven good performance. The numerical approximation of the Jacobian makes convergence a bit slower but it still performs well. It is possible to use an analytic Jacobian, but it was not done in this thesis.

Excitation

To enable identification of joint offsets and kinematic parameters, the data have to be exciting enough. For joint offsets, it means that the joint has to be moved a sufficient amount of radians. It is therefore important to move the joints as much as possible, in as many configurations as possible, to maximize the change to acquire a good identification of the joints. It might, however, not be necessary to move the joints from their minimum angle to their maximum angle. Because the goal is to calibrate the robot such that the desired pose corresponds to the actual pose, it might be sufficient to just move the TCP as much as possible.

Calibration algorithm

The following algorithm is known from [He et al., 2010].

The POE formula gives that the generic forward kinematics of an n-DOF robot manipulator can be written

$$\mathbf{T} = \exp(\widehat{\xi}_1 q_1)\exp(\widehat{\xi}_2 q_2)\ldots\exp(\widehat{\xi}_n q_n)\mathbf{T}_{st}(0) \tag{2.35}$$

where $\mathbf{T}_{st}(0)$ can be seen as the twist of the initial transformation that is given by

$$\mathbf{T}_{st}(0) = \exp(\widehat{\xi}_{st}). \tag{2.36}$$

Linearization of the forward kinematics gives an error model on the form

$$\delta\mathbf{T}\mathbf{T}^{-1} = \left(\frac{\partial T}{\partial \xi}\delta\xi + \frac{\partial T}{\partial \mathbf{q}}\delta\mathbf{q} + \frac{\partial T}{\partial \xi_{st}}\delta\xi_{st}\right)\mathbf{T}^{-1} \tag{2.37}$$

where

$$\xi = [\xi_1, \xi_2, \ldots, \xi_n]^T \in \mathbb{R}^{6n} \tag{2.38}$$

and

$$\mathbf{q} = [q_1, q_2, \ldots, q_n]^T \in \mathbb{R}^n \tag{2.39}$$

A first-order approximation of (2.35) can be written as [He et al., 2010]

$$\delta\mathbf{T}\mathbf{T}^{-1} = \log(\mathbf{T}_a\mathbf{T}_n^{-1}) \in \mathrm{se}(3) \tag{2.40}$$

where $\mathbf{T}_a$ is the actual end-effector pose obtained from measurements and $\mathbf{T}_n$ is the nominal end-effector pose. (2.40) represents the deviation of the actual pose from the nominal pose.

To identify the parameters, the minimization problem

$$\text{minimize} \left\| \delta\mathbf{T}\mathbf{T}^{-1} - \left(\frac{\partial \mathbf{T}}{\partial \xi}\delta\xi + \frac{\partial \mathbf{T}}{\partial \mathbf{q}}\delta\mathbf{q} + \frac{\partial \mathbf{T}}{\partial \xi_{st}}\delta\xi_{st} \right) \mathbf{T}^{-1} \right\|^2 \tag{2.41}$$

has to be solved.

The explicit expression for $\delta\mathbf{T}\mathbf{T}^{-1}$ with the Vee ($\vee$)-operator applied, derived by [He et al., 2010], is now given by

$$\begin{aligned}
&[\delta\mathbf{T}\mathbf{T}^{-1}]^\vee = [(\delta\exp(\widehat{\xi}_1 q_1))\exp(-\widehat{\xi}_1 q_1)]^\vee \\
&+ [\exp(\widehat{\xi}_1 q_1)(\delta\exp(\widehat{\xi}_2 q_2))\exp(-\widehat{\xi}_2 q_2)\exp(-\widehat{\xi}_1 q_1)]^\vee \\
&+ \ldots + \left[\left(\prod_{i=1}^{n-1} \exp(\widehat{\xi}_i q_i) \right) (\delta\exp(\widehat{\xi}_n q_n)\exp(-\widehat{\xi}_n q_n) \times \left(\prod_{i=1}^{n-1} \exp(\widehat{\xi}_i q_i) \right)^{-1} \right]^\vee \\
&+ \left[\left(\prod_{i=1}^{n} \exp(\widehat{\xi}_i q_i) \right) (\delta\exp(\widehat{\xi}_{st}))\exp(-\widehat{\xi}_{st}) \times \left(\prod_{i=1}^{n} \exp(\widehat{\xi}_i q_i) \right)^{-1} \right]^\vee \\
&= [(\delta\exp(\widehat{\xi}_1 q_1))\exp(-\xi_1 q_1)]^\vee \\
&+ Ad(\exp(\widehat{\xi}_1 q_1))[(\delta\exp(\widehat{\xi}_2 q_2))\exp(-\widehat{\xi}_2 q_2)]^\vee \\
&+ \ldots + Ad\left(\prod_{i=1}^{n-1} \exp(\widehat{\xi}_i q_i) \right) [(\delta\exp(\widehat{\xi}_n q_n))\exp(-\widehat{\xi}_n q_n)]^\vee \\
&+ Ad\left(\prod_{i=1}^{n} \exp(\widehat{\xi}_i q_i) \right) [(\delta\exp(\widehat{\xi}_{st})\exp(-\widehat{\xi}_{st})]^\vee
\end{aligned} \tag{2.43}$$

where

$$
\begin{aligned}
&[(\delta \exp(\widehat{\xi}_i q_i)) \exp(-\widehat{\xi}_i q_i)]^{\vee} \\
&\quad = \Bigg(q_i \mathbf{I}_6 + \frac{4 - \theta_i \sin(\theta_i) - 4\cos(\theta_i)}{2\|\omega_i\|^2} \Omega_i \\
&\qquad + \frac{4\theta_i - 5\sin(\theta_i) + \theta_i \cos(\theta_i)}{2\|\omega_i\|^3} \Omega_i^2 \\
&\qquad + \frac{2 - \theta_i \sin(\theta_i) - 2\cos(\theta_i)}{2\|\omega_i\|^4} \Omega_i^3 \\
&\qquad + \frac{2\theta_i - 3\sin(\theta_i) + \theta_i \cos(\theta_i)}{2\|\omega_i\|^5} \Omega_i^4 \Bigg) \delta\xi_i + \xi_i \delta q_i \\
&\quad = \mathbf{A}_i \delta\xi_i + \xi_i \delta q_i
\end{aligned}
\tag{2.44}
$$

and

$$
\begin{aligned}
\Omega_i &= \begin{bmatrix} \widehat{\omega}_i & \widehat{v}_i \\ \mathbf{0} & \widehat{\omega}_i \end{bmatrix} \\
\|\omega_i\| &= (\omega_{1i}^2 + \omega_{2i}^2 + \omega_{3i}^2)^{1/2} \\
\theta_i &= \|\omega_i\| q_i
\end{aligned}
$$

The twist corresponding to the initial transformation matrix is given by

$$
\begin{aligned}
&[(\delta \exp(\widehat{\xi}_{st})) \exp(-\widehat{\xi}_{st})]^{\vee} \\
&\quad = \Bigg(\mathbf{I}_6 + \frac{4 - \theta_{st} \sin(\theta_{st}) - 4\cos(\theta_{st})}{2\theta_{st}^2} \Omega_{st} \\
&\qquad + \frac{4\theta_{st} - 5\sin(\theta_{st}) + \theta_{st} \cos(\theta_{st})}{2\theta_{st}^3} \Omega_{st}^2 \\
&\qquad + \frac{2 - \theta_{st} \sin(\theta_{st}) - 2\cos(\theta_{st})}{2\theta_{st}^4} \Omega_{st}^3 \\
&\qquad + \frac{2\theta_{st} - 3\sin(\theta_{st}) + \theta_{st} \cos(\theta_{st})}{2\theta_{st}^5} \Omega_{st}^4 \Bigg) \delta\xi_i \\
&\quad = \mathbf{A}_{st} \delta\xi_i
\end{aligned}
\tag{2.45}
$$

with

$$
\begin{aligned}
\Omega_{st} &= \begin{bmatrix} \widehat{\omega_{st}} & \widehat{v_{st}} \\ \mathbf{0} & \widehat{\omega_{st}} \end{bmatrix} \\
\theta_{st} &= \|\omega_{st}\| = (\omega_{1st}^2 + \omega_{2st}^2 + \omega_{3st}^2)^{1/2}.
\end{aligned}
$$

Setting

$$\mathbf{J}_i = \begin{cases} [\mathbf{A}_1, \xi_1], & i = 1 \\ Ad\left(\prod_{j=1}^{i-1} \exp(\widehat{\xi}_j q_j)\right)[\mathbf{A}_i, \xi_i], & 1 < i < n+1 \\ Ad\left(\prod_{j=1}^{i-1} \exp(\widehat{\xi}_j q_j)\right)\mathbf{A}_{st}, & i = n+1 \end{cases} \tag{2.46}$$

and

$$\mathbf{J}_{st} = \mathbf{J}_{n+1}, \tag{2.47}$$

(2.43) can be written as

$$\mathbf{y} = \mathbf{J}\mathbf{x} \tag{2.48}$$

where

$$\mathbf{y} = [\delta \mathbf{T}\mathbf{T}^{-1}]^{\vee} \in \mathbb{R}^6 \tag{2.49}$$

$$\mathbf{J} = [\mathbf{J}_1, \mathbf{J}_2, ..., \mathbf{J}_n, \mathbf{J}_{st}] \in \mathbb{R}^{6\times(7n+6)} \tag{2.50}$$

$$\mathbf{x} = [\delta\xi_1, \delta q_1, \delta\xi_2, \delta q_2, ..., \delta\xi_n, \delta q_n, \delta\xi_{st}]^T \in \mathbb{R}^{7n+6} \tag{2.51}$$

$\mathbf{J}$ is the identification Jacobian, $\mathbf{y}$ represents the pose error of the tool frame given in the base frame and $\mathbf{x}$ are the errors in the kinematic parameters. This Jacobian is used for identification of both kinematic parameters and joint offsets. In the case when only kinematic parameters are being identified the Jacobian (2.52) was used

$$\mathbf{J}_i = \begin{cases} \mathbf{A}_1, & i = 1 \\ Ad\left(\prod_{j=1}^{i-1} \exp(\widehat{\xi}_j q_j)\right)\mathbf{A}_i, & 1 < i < n+1 \\ Ad\left(\prod_{j=1}^{i-1} \exp(\widehat{\xi}_j q_j)\right)\mathbf{A}_{st}, & i = n+1 \end{cases} \tag{2.52}$$

and for identification of only offsets the following Jacobian (2.53) was used.

$$\mathbf{J}_i = \begin{cases} \xi_1, & i = 1 \\ Ad\left(\prod_{j=1}^{i-1} \exp(\widehat{\xi}_j q_j)\right)\xi_i, & 1 < i < n+1 \end{cases} \tag{2.53}$$

By measuring m different configurations of the manipulator, (2.48) is given as

$$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_m \end{bmatrix} = \begin{bmatrix} \mathbf{J}_1 \\ \vdots \\ \mathbf{J}_m \end{bmatrix} \mathbf{x} \leftrightarrow \tilde{\mathbf{y}} = \tilde{\mathbf{J}}\mathbf{x}. \tag{2.54}$$

From this relation, $\mathbf{x}$ can be solved for by an iterative least-squares algorithm by use of the pseudo inverse of $\tilde{\mathbf{J}}$ with solution

$$\tilde{\mathbf{x}} = (\tilde{\mathbf{J}}^T\tilde{\mathbf{J}} + \lambda\mathbf{I})^{-1}\tilde{\mathbf{J}}\check{\mathbf{y}}, \tag{2.55}$$

where λ is a scalar. The kinematic parameters are then updated with the kinematic parameter errors as

$$\mathbf{P}_{i+1} = \mathbf{P}_i + \tilde{\mathbf{x}} \tag{2.56}$$

where $\mathbf{P}_0$ are the nominal parameters to start the iteration with (the initial guess).

Identifiability

According to [He et al., 2010], all ξ parameters as well as the base transform ξ_{st} are identifiable when the actual position $\mathbf{T}_a$ is known. It is also possible to identify all ξ parameters and the joint offsets $\delta\mathbf{q}$ provided that there are no more than six DOFs. It is not possible to identify the ξ_{st}- and $\delta\mathbf{q}$-parameters in the same error model because of the fact that the end-effector error would be a linear combination of $\delta\mathbf{q}$ and ξ_{st} and hence they can not appear in the same error model.

Those assumptions require that the actual pose is known and they are valid for one arm only. In the dual-arm implementation in this thesis, the used ”actual” pose in the algorithm is calculated using the erroneous kinematic parameters and joint offsets and will hence not represent the *actual* pose. There is therefore an uncertainty introduced in the algorithm, in contrast to when the known pose is used for the single arm case. This will most definitely affect the identification of the parameters. In the single arm implementation in this thesis, the pose is set to unknown, but with the knowledge that the pose is constant. This will also introduce an uncertainty in the algorithm. Another problem might be that since the arms are clamped together the twists corresponding to the initial poses might be hard to separate in the identification.

2.10 Modification of algorithm

Error handling for dual arms

The error handling in the implementation in this thesis differs from the one in the algorithm described in Section 2.9. The algorithm describes an identification method for one arm with known point and therefore had to be modified for the dual arm implementation. In this thesis, there is no known point for each arm in the dual arm calibration, but it is known that the two TCPs have a common point. In each iteration, the error is calculated as the difference between the two TCPs (compensated for the clamping device). The error is then divided equally between the two arms, so both arms are modified to compensate for the error. If the error was not divided, the correction would therefore be double. A drawback with this approach is that

the algorithm converges slower than it does when only one arm is identified and corrected with the full error.

Damped least squares

Instead of using an ordinary least squares method before updating the parameters, a damped least-squares method was used, see (2.55). Without damping, there is a risk that the parameter update vector $\mathbf{x}$ grows, which might cause the algorithm to become unstable.

Abort criteria

In this thesis, the identification algorithm was aborted either when a specified maximum number of iterations had been executed, or when the norm of the parameter update vector $\mathbf{x}$ from (2.51) was smaller than some specified value. This is because it is desirable to stop the iterations if there no longer is any significant update in each iteration.

When the parameters are updated

In the end of each iteration, there was a check to see if the suggested parameter change would decrease the error. If it would, the parameters would be updated and the damping decreased. If it would not, the parameters would not be updated and the damping would be increased. In some implementations, the criterion was that the new error should be less that 1.5 times the previous error, instead of strictly less than the previous error. It is because the algorithm sometimes seems to work better after a step in the wrong direction. It temporarily increases the error, and can thereafter take a new step that decreases the error.

Offset identification

The offset identification in this thesis is similar to the kinematic parameter identification. However, the nominal twist parameters are used, so the twist parameters are not identified. Therefore only the last part ($\xi_i \delta q_i$) in (2.44) was used (with nominal twist parameters) together with the Jacobian in (2.53).

Conformation of the kinematic parameters

When one value of the twist parameters are changed, the twist parameters have to be adjusted to conform to the definition of a revolute joint [He et al., 2010].

The twist parameters was therefore conformized by using [Yang et al., 2014] :

$$\omega_c = \frac{\omega}{\|\omega\|} \tag{2.57}$$

$$\mathbf{v}_c = \mathbf{v} - \frac{\omega^T \mathbf{v}}{\omega^T \omega} \tag{2.58}$$

Pseudocode

In Algorithms 1, 2, 3, and 4 pseudocode for the implementation of the algorithm in Section 2.9 is presented. The code is inspired by Fredrik Bagge's work, [Bagge, 2016].

In each implementation, a number of logged data points (joint sets) are used. Each joint set contains position values for each joint. The *kinematic cost function* is the difference between the two transformation matrices for each TCP. The *mean transformation matrix* is calculated as a mean value of all joint sets for the translation part. The mean of the rotation is calculated by first converting the rotation matrix $\mathbf{R} = e^{\widehat{\omega}\theta}$ to a twist rotational part ω by taking the logarithm and the using the vee operator from (2.18). A mean is then calculated for all ω and is then converted back to a rotation matrix which is the mean rotation.

Algorithm 1: CalibDual_kinematic

```
Evaluate the initial kinematic error;
for All number of iterations do
    for All joint sets do
        Calculate the forward kinematics for both arms' TCPs with the
          current parameters;
        Calculate the kinematic cost function between the two TCPs;
        Calculate the Jacobian for both arms with current parameters;
    end
    Calculate the kinematic parameter update for both arms separately with a
      damped least-squares method;
    Evaluate the kinematic error;
    if Norm of kinematic parameter error is small then
        break;
    end
    if Error is smaller than before (or in some implementations smaller than
      1.5 times the previous value) then
        Update the kinematic parameters with 50% of the kinematic
          parameter error;
        Conformize the updated kinematic parameters;
        Decrease the damping;
    else
        Increase the damping;
    end
end
```

Algorithm 2: CalibDual_offsets

```
Evaluate the initial kinematic error;
for All number of iterations do
    for All joint sets do
        Calculate the forward kinematics for both arms TCPs with current
          joint values compensated with identified offsets (zero in first
          iteration);
        Calculate the kinematic cost function between the two TCPs;
        Calculate the Jacobian for both arms with joint values compensated
          with identified offsets (zero in first iteration);
    end
    Calculate the joint value update for both arms separately with a damped
      least-squares method;
    Evaluate the kinematic error;
    if Norm of kinematic parameter error is small then
        break;
    end
    if Error is smaller than before (or in some implementations smaller than
      1.5 times the previous value) then
        Update the joint values with the joint value error;
        Decrease the damping;
    else
        Increase the damping;
    end
end
```

Algorithm 3: CalibSingle_offsets

```
Evaluate the initial kinematic error;
for All number of iterations do
    if The fixed point is unknown then
        Calculate a mean transformation matrix with joint values
          compensated with identified offsets (zero in first iteration);
        Orthogonalize the transformation matrix;
    end
    for All joint sets do
        if The fixed point is unknown then
            Calculate the kinematic cost function between the mean
              transformation matrix and current transformation matrix
              calculated with current joint set;
        else
            Calculate the kinematic cost function between the known
              transformation matrix and current transformation matrix
              calculated with current joint set;
        end
        Calculate the Jacobian for the arm with current joint offsets;
    end
    Calculate the joint error for the arm with a damped least-squares method;
    Evaluate the kinematic error;
    if Norm of joint error is small then
        break;
    end
    if Error is smaller than before (or in some implementations smaller than
      1.5 times the previous value) then
        Update the joint values with the joint value error;
        Decrease the damping;
    else
        Increase the damping;
    end
end
```

Algorithm 4: Eval_error

```
0 → rot_error;
0 → trans_error;
for All joint sets do
    Calculate the transformation matrices for both arms;
    Add the norm of the translation difference between the two arms to the
      previous translation difference;
    Transpose one of the rotation matrices, multiply with the other;
    Logarithmize the product and transform the new matrix to a ω vector and
      add the norm to the previous rotation error;
end
Return the sum of transformation error and the sum of rotation error divided
  by the number of joint sets;
```

2.11 Scalability

In identification problems where all geometric lengths are unknown, a problem with scalability can arise [Collin, 2016]. It is because of the fact that the forward kinematics is then calculated solely based on joint angles. If no link length is known, infinitely many solutions can be found for the same set of joint angles. At least one length must therefore be known in the closed kinematic chain in order to break the scalability problem, because the problem with scalability arises when no absolute measurement is available. It is however undesirable to set any length to known (and therefore not identify it), because if the ”known” length is faulty, the identification will lose precision.

3

Simulation and experiments

To verify the algorithms, simulations and experiments were performed. In this chapter a description on how simulation data were generated and how the simulations and experiments were performed is presented.

3.1 Generation of simulation data

To be able to perform simulations, data for the simulations were needed. An iterative least-squares algorithm was implemented to generate joint values corresponding to the poses of the arms, the so called inverse kinematics. The algorithm was implemented using an adaptive damped least-squares algorithm as follows

$$\mathbf{q}_{n+1} = (\mathbf{J}(\mathbf{q}_n)^T \mathbf{J}(\mathbf{q}_n) + \lambda \mathbf{I})^{-1} \mathbf{J}(\mathbf{q}_n)\mathbf{y}. \tag{3.1}$$

In (3.1), $\mathbf{y}$ is the error between the desired pose and the iteratively updated pose, starting from the intial guess expressed as twist coordinates, and the Jacobian $\mathbf{J}(\mathbf{q})$ is defined as [Murray et al., 1994]

$$\mathbf{J} = \begin{bmatrix} \xi_1 & \xi_2' \dots \xi_n' \end{bmatrix} \tag{3.2}$$

where ξ_i' is given by

$$\xi_i' = Ad_{(e^{\hat{\xi}_1\theta_1 \dots e^{\hat{\xi}_{i-1}\theta_{i-1}}})}\xi_i \tag{3.3}$$

and $\lambda \in \mathbb{R}$ is a damping factor which is changed depending on how the error changes. The factor is multiplied by ten if the current error is bigger than the last and divided by ten if the current error is smaller than the last.

3.2 Base transformation

Since both arms are to be connected when performing the calibration, both arms have the same base coordinate frame. This simplifies the comparison between the poses of the two arms which is used in the algorithms. The transformation from

the common base frame to each arm base needs to be known. For YuMi, the base transform calculation described in Section 2.4 was performed with data from RobotStudio, which should lead to a very accurate base transformation. Since Frida is an early prototype there was reason to assume that it most likely differ a bit from YuMi in its specifications, so a base transformation earlier derived by a former PhD student[1] was used for Frida.

3.3 Clamped single arm, Frida

One fixed point

Idea The idea behind the clamping method is to fix the TCP at a desired pose, and since the robot is redundant it has the possibility to reach the position in many different configurations. It can be compared to pressing the palm of the hand towards a table, and moving the arm around in different configurations. The human arm, including the shoulder, elbow and wrist, has seven DOFs. If the TCP (or the palm of the hand) is positioned in a fixed point, it is only possible to move the elbow in space. This situation is the same for the robot when attached in a fixed pose at a fixed point. If there is any offset in the joints, the registered pose will differ from the actual pose at the fixed point, even though the TCP has the same pose. By using, for example, least-squares identification, the joint offsets can be calculated.

Simulation Via the Python implementation of the inverse kinematics, several configurations (sets of joint values) were generated with the requirement that the TCP should have the same pose, but with different joint angles. A known offset was added to the joint values and via the known transformation matrix and the joint sets, the joint offsets were calculated with the implementation of the algorithm in Section 2.9. In the first simulation, the points were assumed to be known, making the simulation not fully realistic, but tested how well the joint offset calibration worked.

To make a more realistic experiment, the points were now set to be unknown. The joint sets from the above simulation were used, and the same offset was added. In Algorithm 3, transformation matrices were calculated for all joint values, and a mean was used as the actual measurement in the algorithm. In each loop, new transformation matrices were calculated, corrected with the newly calculated offsets. With this approach, the different poses will converge to one point and the offsets for that point is the result.

Experiment For the experiment, two tool changers were mounted on a thick metal plate with a constant distance of approximately 5 cm, see Figure 3.1. The plate with the tool changer was then attached to the robot, because the robot itself was not that firmly attached to the table, and therefore the most stable point was on the robot. The tool changer plate was possible to mount on both sides of the robot, so both

[1] Andreas Stolt.

arms could be fixed in the same way. One arm was then attached to the tool changer closest to the robot, see Figure 3.2, and the arm was then manually moved as much as possible after the brakes were released. Since the end-effector was attached to the tool changer it kept a fixed pose during the movement. The result was several configurations (sets of joint values), that corresponded to the fixed pose. Because of joint limitations, not all configurations were reachable, so the arm had to be detached, reoriented, and then attached again. It was possible to attach the arm in two different ways, one where the elbow was moving in a semicircle inwards towards the body and one where it was moving outward towards the body, resulting in two different obtained data sets. This increased the number of configurations and also increased the total joint excitation. The same procedure was carried out on both arms. The distance from the robot base to the tool changers was measured to get a rough estimation if the result was reasonable.

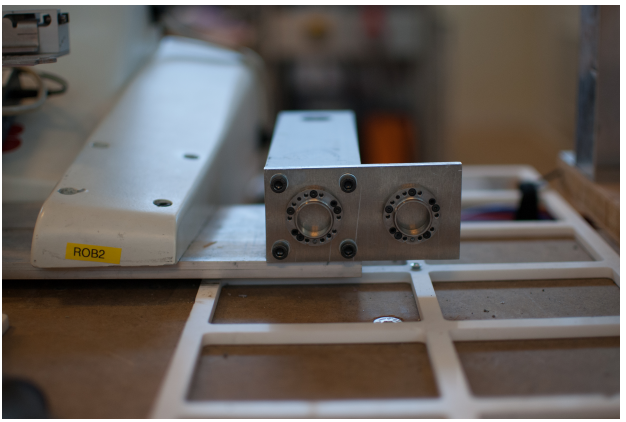

Figure 3.1 The clamping device with tool changers (Schunk, model MWS030) mounted on Frida's left side.

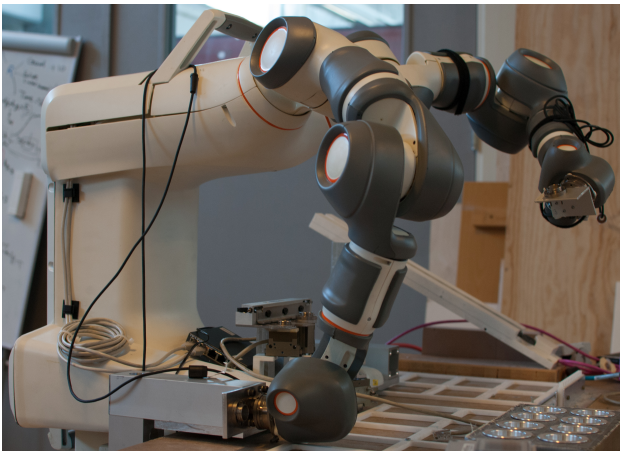

Figure 3.2 Frida's right arm at the clamping device.

Several fixed points

Idea Using only one point, there is a problem with identifying the offsets for joint one and joint seven, which can be seen in Table 4.2. An offset on the first joint will not affect the possibility to repeat a pose in a certain point but it will affect where that point is located, and an offset on joint seven will not be distinguishable from a rotation of the fixed point. This can be read more about in Section 5.1. The idea behind using several fixed points with known transformation in between, instead of only one point is to improve the identifiability of joint one and joint seven. This is done by addressing the redundancy problem where an offset on joint one could be compensated for by moving the clamping point, and an offset on joint seven could be compensated by rotating the fixed point.

Simulation Similar to the one fixed point case, data for several points spread throughout the room were generated randomly. Restrictions in x, y, and z coordinates were introduced to keep the points in the work space of the arm. The randomized 3x3 rotation matrix was orthogonalized to represent a valid rotation. One point was chosen to be the original point and the transformation between this point and the others were calculated using the relationship

$$\mathbf{T}_{trans} = \mathbf{T}^{-1}_{additional}\mathbf{T}_{original} \tag{3.4}$$

where $\mathbf{T}_{trans}$ is the transformation between the original point and the additional point and $\mathbf{T}_{original}$ and $\mathbf{T}_{additional}$ are the forward kinematic maps for the original and additional points respectively. The same offsets were added to all points and the result is shown in Figure 3.3. The calculated transformation matrices for each point

were used to move the four additional points to the original point using

$$\mathbf{T}_{original} = \mathbf{T}_{additional}\mathbf{T}_{trans} \tag{3.5}$$

which resulted in Figure 3.4 (note the different scaling). The identification was then performed on the total collection of data sets. The cost function that was minimized in this identification is given by

$$\mathbf{J} = \sum f_i \cdot f_i^T \tag{3.6}$$

and is a sum of the error between the forward kinematics of all points in the collection of data sets and a twist representing a homogeneous transformation matrix (point) that is to be optimized to make the cost function as small as possible. In (3.6), the row-vector f_i is given by

$$f_i = (\log(\mathbf{T}^i_{fkin}(\mathbf{T}_{twist})^{-1}))^\wedge \tag{3.7}$$

and represents the twist coordinates representing the error in transformation between all the measured poses ($\mathbf{T}^i_{fkin}$) and a transformation matrix calculated by using the exponential of a twist ($\mathbf{T}_{twist}$). The optimization variables are the joint offsets and the twist coordinates corresponding to $\mathbf{T}_{twist}$. The index i is to denote the measured poses.

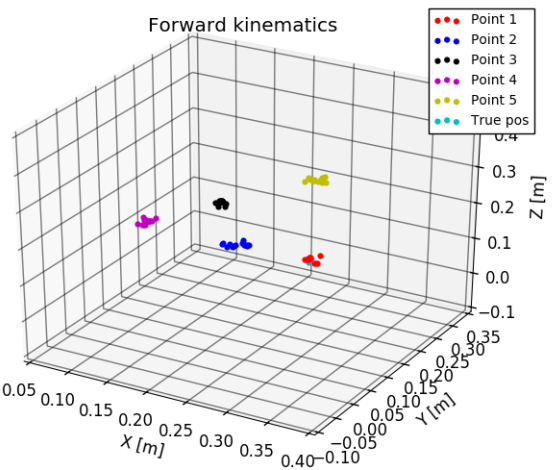

Figure 3.3 Spread of the points before they are transformed to the original point.

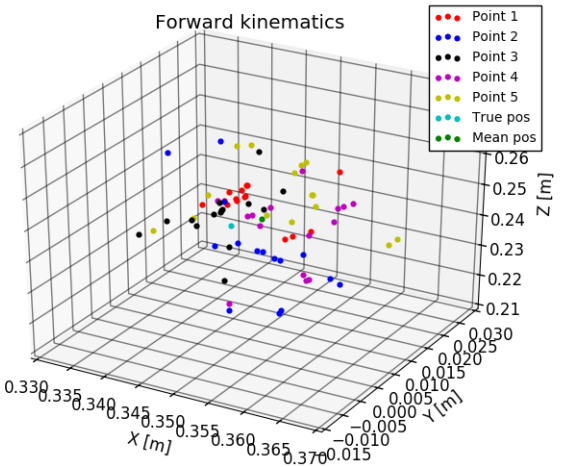

Figure 3.4 Spread of the points after they are transformed to the original point.

Experiment This method was only evaluated in simulations.

Logging

All logging was executed using ExtCtrl with no down-sampling (0.004032 s between samples). The data from ExtCtrl used motor radians as measurement unit and used the joint numbering for Frida (1,2,7,3,4,5,6 from the arm base). The data were then down-sampled in Python, converted to arm radians by dividing the motor radians with the gear ratios given in Table 3.1 and using joint numbering from one to seven counting from the base. The odd gear ratio for joint five is because the joint is located in the tilt house, which means that if joint five rotates a full revolution, the engine will rotate a full revolution. This leads to one extra (101 instead of 100) in gear ratio compared to the other joints.

Validation

To validate the identified offsets, they were added to all joint sets and the forward kinematics was then calculated. If all joint offsets were successfully calculated, all joint sets will give the same pose. This was done offline with the logged data.

Table 3.1 Gear ratio from motor radians to arm radians.

Joint	Gear ratio
1	100
2	100
3	-100
4	100
5	-101
6	100
7	100

3.4 Dual arm calibration, Frida

Idea

The idea with the dual arm calibration is similar to the one behind the fixed point clamping method, but instead of a fixed pose there is a fixed transformation between the two TCPs. If the robot is uncalibrated, there will be conflicting transformations between the two TCPs for the different measured poses (measured with the logged joint values). By using the least-squares method as described in Algorithm 1, an attempt to identify the errors in the ξ-parameters was carried out, as well as joint offset identification.

Simulation

Before the actual experiment, simulations were carried out. A fixed transformation represented as a twist was added to the right arm, corresponding to the clamping device between the two arms. The clamping device between the two end-effectors can be seen as an extension of the last link of the right arm. The z-axis pointing from the clamping device (on the right arm) into the left arm's TCP was rotated around the y-axis, to be pointing in the same direction as the z-axis of the left arm. By doing that, the optimization problem was simplified to be a problem with minimum when the two end-effectors (the original one from the left arm and the new one from the end of the clamping device on the right arm) to have the same position and the same orientation. Random poses were generated for the right arm (with the clamping device added), and via inverse kinematics, poses for the left arm were calculated to match the transformation corresponding to the right arm and the clamping device (with the previously described rotated z-axis).

Parameter identification To simulate uncertainties in the robot parameters, small variations were added to the nominal ξ-parameters before Algorithm 1 was used. Since only kinematic parameters were to be identified the Jacobian in (2.52) was used. The variations were added by randomizing a number using NumPy's *random.uniform-function* in the interval [-1,1] multiplied by 0.01 m for the linear components and $2.4\pi/180$ rad for the angular components. These variations gener-

ated a total error of up to about 0.05 m in translation and 0.15 radians in orientation for the TCP.

Offset identification To simulate offsets, a predetermined offset was added to each joint with absolute value in the range 0.0 to 0.07 radians. The algorithm in Section 2.9 was used, where only $\boldsymbol{\delta}\mathbf{q}$ was identified, and the Jacobian was calculated according to (2.53) in difference to the case of kinematic parameter identification. Small uncertainties were then added to the ξ-parameters to test the identification under more realistic circumstances and the simulation was redone.

Parameter and offset calibration There is an overlap in the joint offset identification and the kinematic parameter identification. A joint offset can be compensated by either adding the identified joint offset, or changing the kinematic parameters including the parameters corresponding to $\mathbf{T}_{st}$. Both identification methods will therefore individually solve the problem. Since both errors in kinematic parameters and offsets were assumed, first a joint offset calibration was executed for some iterations, followed by a kinematic parameter calibration with the updated joint values retrieved from the offset calibration.

Experiment

To automate the experiment procedure, the movements were written in RAPID code, tested in RobotStudio and then uploaded to the robot. A start position was chosen, and MultiMove (see Section 1.7) was then used to fix the two arms' relative position and orientation. The left arm was set to follow the right arm. Different configurations were then programmed in RobotStudio (for the YuMi robot) and then uploaded to the Frida robot. The positions were chosen to maximize the angular movements of each joint. To confirm that reasonable configurations were chosen, a first test was executed with the poses, but where the two arms were not connected. After the different poses were confirmed to be reasonable, the arms were connected with the clamping device, see Figure 3.5. At first, a relatively large value for Soft Servo (see Section 1.7) was used, and successively decreased to a value where all configurations were possible to reach. A too large Soft Servo value results in that the motor force is too weak and the robot is unable to move, and for a too small value there is a risk that the motors will get damaged because of the high force from the joint motors. Because the YuMi robot in RobotStudio and the Frida robot does not correspond fully, manual correction of the poses had to be done by manually moving the clamped arms and then saving the position. Because of the joint offsets and the kinematic parameter errors, the two TCPs would not follow each other perfectly.

Figure 3.5 The dual arm clamping device with toolchangers on a 90 degree bracket.

It was difficult to write a RAPID program that made a big motion since it was time consuming and quite a lot of work to manually change the positions for that many points, which was needed because of the problems mentioned above. Instead, the connected arms were moved manually as much as possible after the brakes for both arms were released to get a good data set. That data was then used in the identification experiments.

Logging

The data was logged in the same manner as described in Section 3.3.

Identify the clamping device

To run the optimization algorithm, the transformation representing the clamping device must be known. To find the transformation, the following relationship was used:

$$\mathbf{T}_l \mathbf{T}_{cd} = \mathbf{T}_r \Rightarrow \mathbf{T}_{cd} = \mathbf{T}_l^{-1} \mathbf{T}_r \tag{3.8}$$

where $\mathbf{T}_l$ is the transformation matrix from the base coordinate system to the TCP for the left arm, $\mathbf{T}_r$ is the transformation matrix from the base coordinate system to the TCP for the right arm and $\mathbf{T}_{cd}$ is the transformation matrix for the clamping device that was used. This is done for a lot of samples when moving in the dual arm configuration. $\mathbf{T}_r$ and $\mathbf{T}_l$ is calculated for every joint configuration in each sample, and the mean of all $\mathbf{T}_{cd}$ is calculated from each sample. Every element in the mean $\mathbf{T}_{cd}$ was calculated as a mean value of the corresponding element in each $\mathbf{T}_{cd}$ from all samples.

The sensitivity of estimation errors for the clamping device was calculated via the following steps.

1. Simulated data for dual arm with known joint offset and known clamping device.
2. Identify the joint offset via the joint offset identification algorithm.
3. Subtract the vector with identified joint offsets from the vector with known joint offsets element-wise and calculate the norm of the vector.
4. Change the transformation matrix for the clamping device to simulate uncertainties.
5. Once again, identify the joint offset.
6. Once again, calculate the norm for the known joint offset with the identified joint offset subtracted.
7. Compare the norm from Step 3 with the norm from Step 6.

These steps will give an indication how much a given uncertainty in the clamping device affects the joint offset identification. If the norm between Step 3 and Step 6 increases, it is an indication that the joint offset identification is affected.

4

Results

The following simulations and experiments have been performed:

- One arm
  - One fixed point, offset identification.
  - Several fixed points, offset identification.
  - Connected to a calibrated arm, kinematic parameter identification.
- Dual arm
  - Offset identification.
  - Kinematic parameter identification.
  - Joint offsets identification followed by kinematic parameter identification.

4.1 Simulation results

In this section, the calibration methods are evaluated on simulated data. When iterations are mentioned on the format x(y), x is the total number of iterations and y is the number of iterations where the error has been decreased (or is less than 1.5 times the previous error in some implementations). The values are only changed when the error decreases, as mentioned in Section 2.10.

The cost function for the one arm, one fixed point was calculated as

$$\|y - Ax\| \tag{4.1}$$

where y is the difference between the transformation matrix for the fixed point (sometimes calculated as a mean) and the transformation matrix which is calculated with the forward kinematics and the updated joint offsets and kinematic parameters. A is the jacobian and x is the error vector.

For the case with one arm but several fixed points, the cost function is calculated as the sum of the distance between the points calculated with the forward kinematics for all joint values after they are moved to the original point and the mean of all these points. The details can be seen in Section 3.3.

For the dual arm offset cases, the cost function is calculated as (4.1) for each arm, but y is calculated as the difference between the two transformation matrices for each arm, A is the jacobian for each arm and x is the error vector for each arm.

Frida one arm clamping

Known point In Table 4.1, the result from the iterations with a known point is presented, that is, when the pose of the arm is known. In Figure 4.1, the original uncalibrated data points are presented in black and the calibrated points in red. The final cost function was 1.23e-08.

Table 4.1 Known point identifying offsets in simulation from 3(3) iterations. All values are in radians.

Joint	Nominal offset	Actual values	Identified values
1	$\begin{pmatrix}0\\0\\0\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0.02000\\-0.03000\\-0.02000\\0.03000\\0.01500\\0.04000\\0.01800\end{pmatrix}$	$\begin{pmatrix}0.02000\\-0.03000\\-0.02000\\0.03000\\0.01500\\0.04000\\0.01800\end{pmatrix}$

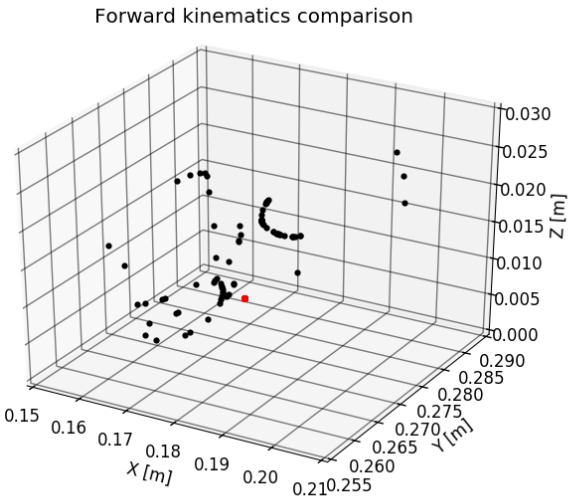

Figure 4.1 Joint offset identification with known point. Black is the uncalibrated data points, red the calibrated data points, which all coincide in one point.

Unknown point In Table 4.2 the result from the joint offset identification for all joints for the left arm with unknown point is presented. In Figure 4.2, the forward kinematics before calibration (black) and after (red) are presented. The final cost function was 0.00883.

Table 4.2 Joint offset identification with unknown clamping point for 10(10) iterations. All values are in radians.

Joint	Nominal offset	Actual values	Identified values
1	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}0.02000\end{pmatrix}$	$\begin{pmatrix}-0.01528893\end{pmatrix}$
2	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}-0.03000\end{pmatrix}$	$\begin{pmatrix}-0.0298328\end{pmatrix}$
3	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}-0.02000\end{pmatrix}$	$\begin{pmatrix}-0.01723198\end{pmatrix}$
4	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}0.03000\end{pmatrix}$	$\begin{pmatrix}0.02995881\end{pmatrix}$
5	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}0.01500\end{pmatrix}$	$\begin{pmatrix}0.01839823\end{pmatrix}$
6	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}0.04000\end{pmatrix}$	$\begin{pmatrix}0.03976109\end{pmatrix}$
7	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}0.00180\end{pmatrix}$	$\begin{pmatrix}-0.0179892\end{pmatrix}$

Table 4.3 The actual pose and the identified pose of the arm from the multiple point simulation.

Actual pose				Identified pose			
1.0000	0.0000	0.0000	0.3500	0.9999	−0.0091	0.0067	0.3495
0.0000	1.0000	0.0000	0.0000	0.0092	0.9998	−0.0155	0.0052
0.0000	0.0000	1.0000	0.2500	−0.0066	0.0155	0.9999	0.2470
0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	1.0000

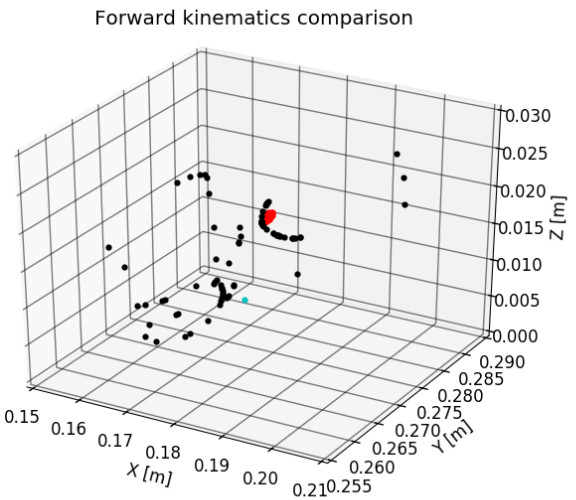

Figure 4.2 Joint offset identification with unknown point for the left arm. Uncalibrated data points are presented in black, calibrated data points in red. The cyan colored point corresponds to the actual position.

Several points In Figure 4.3, the result from the offset calibration can be seen. All moved points now correspond to the same point. The identified twist represented as a transformation matrix (pose) and the actual pose can be seen in Table 4.3. In Table 4.4 the identified joint offsets can be seen. The norm of the translation error between the identified point and the actual point is 0.0060 m. The final value of the cost function is 7.913e-14.

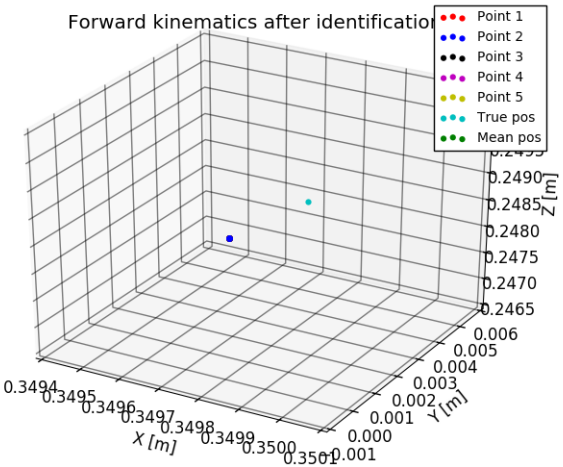

Figure 4.3 Forward kinematics comparison after calibration, where all points coincide at the blue dot.

Table 4.4 Identified offsets from the multiple point simulation.

Joint	Nominal offset	Actual values	Identified values
1	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}0.0200\end{pmatrix}$	$\begin{pmatrix}-0.0008\end{pmatrix}$
2	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}-0.0300\end{pmatrix}$	$\begin{pmatrix}-0.0300\end{pmatrix}$
3	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}-0.0200\end{pmatrix}$	$\begin{pmatrix}-0.0200\end{pmatrix}$
4	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}0.0300\end{pmatrix}$	$\begin{pmatrix}0.0300\end{pmatrix}$
5	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}0.0150\end{pmatrix}$	$\begin{pmatrix}0.0150\end{pmatrix}$
6	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}0.0300\end{pmatrix}$	$\begin{pmatrix}0.0300\end{pmatrix}$
7	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}0.0180\end{pmatrix}$	$\begin{pmatrix}0.0180\end{pmatrix}$

Dual arm

Joint offsets In Table 4.5, the result from the simulation with dual arm and known clamping device is presented. The final value for the cost function for left arm is: 0.07133 and the cost function for right arm is 0.0640.

A similar simulation was performed where only identification of offsets in joint one in both arms was performed. The result is shown in Table 4.6. The cost function values were 1.0866e-07 and 1.6371e-07 for the left and right arm respectively.

In Table 4.7, the result from the simulation with dual arm and known clamping device is presented with the criterion for applying the update relaxed with factor

Table 4.5 Joint offset identification for dual arm in simulation for 10(2) iterations. All values are in radians.

Joint	Nominal offset	Actual offsets	Identified left	Identified right
1	0	0.02000	0.00706375	0.02092654
2	0	−0.03000	−0.02981544	−0.02864519
3	0	−0.02000	−0.02889904	−0.02121532
4	0	0.03000	0.03062328	0.0330501
5	0	0.01500	0.00603746	0.00998059
6	0	0.04000	0.03830235	0.03322925
7	0	0.0180	0.01648792	0.01583548

Table 4.6 Joint offset identification for joint one for dual arm in simulation for 50(50) iterations. All values are in radians.

Joint	Nominal offset	Actual offsets (L/R)	Identified (L/R)
1	0	0.0200/ − 0.0300	0.0200/ − 0.0300
2	0	0.0000	0.0000
3	0	0.0000	0.0000
4	0	0.0000	0.0000
5	0	0.0000	0.0000
6	0	0.0000	0.0000
7	0	0.0000	0.0000

1.5. The final value for the cost function for the left arm is 0.0386 and for the right arm is 0.00307.

Table 4.7 Joint offset identification for dual arm in simulation for 10(10) iterations, with relaxed requirement for applying the update with factor 1.5. All values are in radians.

Joint	Nominal offset	Actual offsets	Identified left	Identified right
1	0	0.02000	0.01166092	0.01972847
2	0	−0.03000	−0.02951589	−0.02992553
3	0	−0.02000	−0.02491096	−0.02038856
4	0	0.03000	0.03024651	0.02995226
5	0	0.01500	0.00992934	0.01458749
6	0	0.04000	0.03971442	0.04001757
7	0	0.0180	0.01682413	0.01787835

Dual arm where one arm is considered known In Table 4.8, the identified parameters from simulations can be seen with the nominal and actual parameters for comparison. The simulations were performed with one arm considered to be known and with different kinds of errors added to the model. Firstly, the simulation was

performed with errors only added to the kinematic parameters. In the next step, offsets were added to the joint values in addition to the kinematic parameter errors. In the last step measurement noise was added as well. The errors in translation and rotation from the simulations can be seen in Figures 4.4 and 4.5.

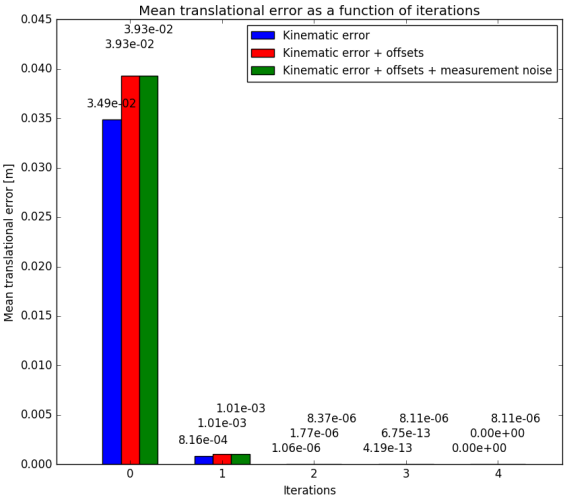

Figure 4.4 Position error as a function of iterations for different types of added errors.

Table 4.8 Nominal, actual, and identified parameters for one arm in dual arm simulation, where one arm is assumed to be known and used as measurement.

Twist	Nominal parameters	Actual parameters	Identified parameters Kinematic errors	Identified parameters Kinematic errors + offsets	Identified parameters Kinematic errors + offsets + measurement noise
ξ_1	$\begin{pmatrix}0\\0\\0\\0\\0\\1\end{pmatrix}$	$\begin{pmatrix}-0.0051\\-0.0022\\0.0001\\0.0095\\0.0253\\0.9996\end{pmatrix}$	$\begin{pmatrix}-0.0051\\-0.0022\\0.0001\\0.0095\\0.0253\\0.9996\end{pmatrix}$	$\begin{pmatrix}-0.0051\\-0.0022\\0.0001\\0.0095\\0.0253\\0.9996\end{pmatrix}$	$\begin{pmatrix}-0.0051\\-0.0022\\0.0001\\0.0095\\0.0253\\0.9996\end{pmatrix}$
ξ_2	$\begin{pmatrix}-0.11\\-0.0\\-0.03\\0\\1\\0\end{pmatrix}$	$\begin{pmatrix}-0.1052\\0.0022\\-0.0228\\0.0283\\0.9990\\-0.0332\end{pmatrix}$	$\begin{pmatrix}-0.1052\\0.0022\\-0.0228\\0.0283\\0.9990\\-0.0332\end{pmatrix}$	$\begin{pmatrix}-0.1052\\0.0043\\-0.0227\\0.0483\\0.9983\\-0.0334\end{pmatrix}$	$\begin{pmatrix}-0.1052\\0.0043\\-0.0227\\0.0483\\0.9983\\-0.0334\end{pmatrix}$
ξ_3	$\begin{pmatrix}0.0\\-0.0\\0.0\\0\\0\\1\end{pmatrix}$	$\begin{pmatrix}-0.0021\\-0.0027\\0.0002\\0.0370\\0.0397\\0.9985\end{pmatrix}$	$\begin{pmatrix}-0.0021\\-0.0027\\0.0002\\0.0370\\0.0397\\0.9985\end{pmatrix}$	$\begin{pmatrix}-0.0019\\0.0003\\0.0001\\0.0672\\0.0376\\0.9970\end{pmatrix}$	$\begin{pmatrix}-0.0019\\0.0003\\0.0001\\0.0672\\0.0376\\0.9970\end{pmatrix}$
ξ_4	$\begin{pmatrix}-0.3565\\0.0\\0.0405\\0\\1\\0\end{pmatrix}$	$\begin{pmatrix}-0.3609\\-0.0049\\0.0313\\-0.0154\\0.9996\\-0.0219\end{pmatrix}$	$\begin{pmatrix}-0.3609\\-0.0049\\0.0313\\-0.0154\\0.9996\\-0.0219\end{pmatrix}$	$\begin{pmatrix}-0.3591\\-0.0046\\0.0390\\-0.0150\\0.9997\\-0.0194\end{pmatrix}$	$\begin{pmatrix}-0.3591\\-0.0046\\0.0390\\-0.0150\\0.9997\\-0.0194\end{pmatrix}$
ξ_5	$\begin{pmatrix}0.0\\0.397\\-0.0\\1\\0\\0\end{pmatrix}$	$\begin{pmatrix}-0.0091\\0.3964\\-0.0027\\0.9996\\0.0230\\0.0134\end{pmatrix}$	$\begin{pmatrix}-0.0091\\0.3964\\-0.0027\\0.9996\\0.0230\\0.0134\end{pmatrix}$	$\begin{pmatrix}-0.0089\\0.3946\\-0.0018\\0.9997\\0.0226\\0.0132\end{pmatrix}$	$\begin{pmatrix}-0.0089\\0.3946\\-0.0018\\0.9997\\0.0226\\0.0132\end{pmatrix}$
ξ_6	$\begin{pmatrix}-0.3835\\0.0\\0.3055\\0\\1\\0\end{pmatrix}$	$\begin{pmatrix}-0.3753\\-0.0056\\0.3146\\-0.0018\\0.9999\\0.0157\end{pmatrix}$	$\begin{pmatrix}-0.3753\\-0.0056\\0.3146\\-0.0018\\0.9999\\0.0157\end{pmatrix}$	$\begin{pmatrix}-0.3737\\-0.0015\\0.3224\\-0.0012\\1.0000\\0.0032\end{pmatrix}$	$\begin{pmatrix}-0.3737\\-0.0015\\0.3224\\-0.0012\\1.0000\\0.0032\end{pmatrix}$
ξ_7	$\begin{pmatrix}0.0\\0.4105\\-0.0\\1\\0\\0\end{pmatrix}$	$\begin{pmatrix}-0.0144\\0.4133\\0.0088\\0.9988\\0.0356\\-0.0331\end{pmatrix}$	$\begin{pmatrix}-0.0144\\0.4133\\0.0088\\0.9988\\0.0356\\-0.0331\end{pmatrix}$	$\begin{pmatrix}-0.0138\\0.4025\\0.0095\\0.9994\\0.0344\\-0.0034\end{pmatrix}$	$\begin{pmatrix}-0.0138\\0.4025\\0.0095\\0.9994\\0.0344\\-0.0034\end{pmatrix}$
ξ_{st}	$\begin{pmatrix}-0.0573\\0.0\\0.5874\\0.0\\1.5707\\0.0\end{pmatrix}$	$\begin{pmatrix}-0.0605\\0.0033\\0.5796\\-0.0280\\1.5750\\-0.0296\end{pmatrix}$	$\begin{pmatrix}-0.0605\\0.0033\\0.5796\\-0.0280\\1.5750\\-0.0296\end{pmatrix}$	$\begin{pmatrix}-0.0448\\-0.0001\\0.5822\\-0.0507\\1.5439\\-0.0542\end{pmatrix}$	$\begin{pmatrix}-0.0448\\-0.0001\\0.5822\\-0.0507\\1.5439\\-0.0542\end{pmatrix}$

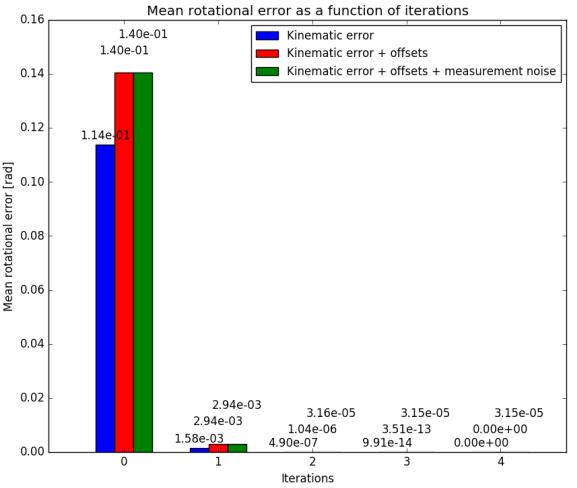

Figure 4.5 Rotational error as a function of iterations for different types of added errors.

Dual arm calibration where both arms are unknown In Figures 4.6 and 4.7, results from simulations of a dual arm calibration can be seen. In the simulation, errors in the kinematic parameters have been introduced and the mean errors during the course of the calibration are presented. The resulting identified parameters for the left and right arm can be seen in Table 4.9.

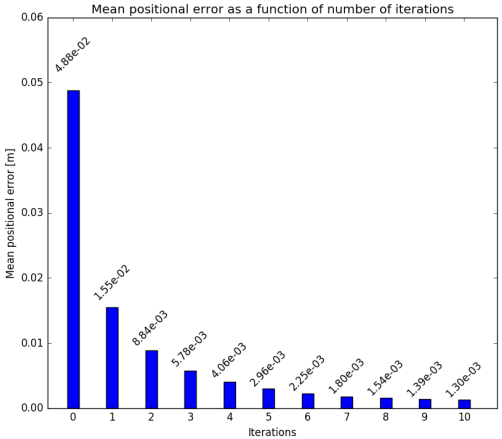

Figure 4.6 How the error in translation decreases in the algorithm.

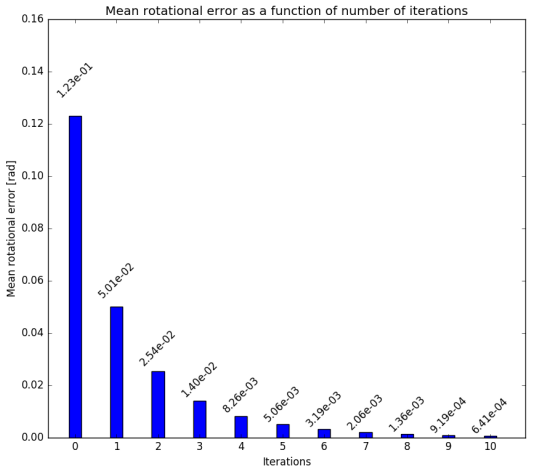

Figure 4.7 How the error in rotation decreases in the algorithm.

Table 4.9 Identified parameters in dual arm calibration, where only errors in kinematic parameters were introduced.

Twist	Right arm actual parameters	Right arm identified parameters	Left arm actual parameters	Left arm identified parameters
ξ_1	$\begin{pmatrix}-0.0007\\-0.0002\\-0.0000\\-0.0227\\0.0184\\0.9996\end{pmatrix}$	$\begin{pmatrix}-0.0040\\-0.0043\\-0.0001\\-0.01268\\0.0176\\0.9998\end{pmatrix}$	$\begin{pmatrix}0.0090\\-0.0019\\-0.0001\\0.0177\\0.0076\\0.9998\end{pmatrix}$	$\begin{pmatrix}0.0036\\-0.0016\\-0.0000\\0.0059\\0.066\\1.0000\end{pmatrix}$
ξ_2	$\begin{pmatrix}-0.1000\\-0.0018\\-0.0312\\-0.0096\\0.9996\\-0.0270\end{pmatrix}$	$\begin{pmatrix}-0.1024\\-0.0040\\-0.0351\\-0.0297\\0.9992\\-0.0261\end{pmatrix}$	$\begin{pmatrix}-0.1191\\0.0005\\-0.0341\\-0.0016\\0.9998\\0.0200\end{pmatrix}$	$\begin{pmatrix}-0.1167\\-0.0016\\-0.0362\\-0.0208\\0.9995\\0.0220\end{pmatrix}$
ξ_3	$\begin{pmatrix}-0.0073\\0.0058\\0.0002\\0.0121\\-0.0153\\0.9998\end{pmatrix}$	$\begin{pmatrix}-0.0107\\0.0107\\0.0004\\0.0231\\-0.0152\\0.9996\end{pmatrix}$	$\begin{pmatrix}0.0043\\-0.0041\\-0.0001\\0.0071\\-0.0271\\0.9996\end{pmatrix}$	$\begin{pmatrix}-0.0014\\-0.0038\\-0.0001\\-0.0038\\-0.0282\\0.9996\end{pmatrix}$
ξ_4	$\begin{pmatrix}-0.3482\\0.0119\\0.0472\\0.0302\\0.9991\\-0.0287\end{pmatrix}$	$\begin{pmatrix}-0.3520\\0.0064\\0.0459\\0.0144\\0.9995\\-0.0291\end{pmatrix}$	$\begin{pmatrix}-0.3506\\0.0046\\0.0306\\0.0102\\0.9994\\-0.0328\end{pmatrix}$	$\begin{pmatrix}-0.3507\\-0.0008\\0.0265\\-0.0047\\0.9995\\-0.0316\end{pmatrix}$
ξ_5	$\begin{pmatrix}-0.0022\\0.3961\\0.0060\\1.0000\\0.0054\\0.0085\end{pmatrix}$	$\begin{pmatrix}-0.0081\\0.4012\\0.0074\\0.9998\\0.0203\\-0.0021\end{pmatrix}$	$\begin{pmatrix}0.0129\\0.3874\\-0.0093\\0.9994\\-0.0334\\-0.0014\end{pmatrix}$	$\begin{pmatrix}0.0073\\0.3874\\-0.0040\\0.9998\\-0.0186\\0.0104\end{pmatrix}$
ξ_6	$\begin{pmatrix}-0.3846\\0.0016\\0.3058\\0.0370\\0.9985\\0.0412\end{pmatrix}$	$\begin{pmatrix}-0.3864\\-0.0025\\0.3073\\0.0220\\0.9991\\0.0358\end{pmatrix}$	$\begin{pmatrix}-0.3750\\-0.0103\\0.2956\\-0.0107\\0.9997\\0.0213\end{pmatrix}$	$\begin{pmatrix}-0.3783\\-0.0150\\0.2932\\-0.0259\\0.9995\\0.0177\end{pmatrix}$
ξ_7	$\begin{pmatrix}-0.0014\\0.4025\\0.0008\\0.9998\\0.0035\\0.0217\end{pmatrix}$	$\begin{pmatrix}-0.0075\\0.4075\\0.0021\\0.9998\\0.0183\\0.0107\end{pmatrix}$	$\begin{pmatrix}0.0011\\0.4159\\0.0069\\0.9998\\-0.0024\\-0.0171\end{pmatrix}$	$\begin{pmatrix}-0.0050\\0.4162\\0.0120\\0.9999\\0.0121\\-0.0054\end{pmatrix}$
ξ_{st}	$\begin{pmatrix}-0.0552\\-0.0043\\0.5831\\0.0220\\1.6048\\-0.0163\end{pmatrix}$	$\begin{pmatrix}-0.04811\\-0.0060\\0.5798\\0.0055\\1.5896\\-0.0076\end{pmatrix}$	$\begin{pmatrix}-0.0592\\0.0010\\0.5935\\0.0301\\1.5931\\0.0247\end{pmatrix}$	$\begin{pmatrix}-0.0562\\-0.0035\\0.5911\\0.0161\\1.5557\\0.0306\end{pmatrix}$

Identifiability

Jacobian analysis The Jacobian analysis result is presented in Table 4.10, where n is the number of DOF and N corresponds to the number of measured poses. N has a minimum value to make the Jacobian have more rows than columns, to be able to evaluate if it has full column rank or not.

According to Table 4.10, there will be a linear dependency when identifying

all ξ-parameters and $\delta\mathbf{q}$, but not when identifying all ξ-parameters and $\mathbf{T}_{st}$. The non-identifiable parameter is δq_3.

According to the linear parameter identification, there is a linear dependency between the parameters δq_1, δq_3, $\delta v_{2,2}$ $\delta\omega_{2,1}$. The subscript (x,y) means that it belongs to the x:th twist and is the corresponding twist coordinate's y:th $\mathbf{v} \in \mathbb{R}^3$ or $\omega \in \mathbb{R}^3$ parameter.

Table 4.10 QR-decomposition of the Jacobian for one arm. The number within parenthesis is the theoretical full rank.

Data set	Jacobian dimensions	Jacobian rank	Zero diagonal elements in R
$\xi + \mathbf{T}_{st}$	$6N$x($6n$+6)	48 (48)	0
$\xi + \delta\mathbf{q}$	$6N$x$7n$	48 (49)	1

Values after many iterations

In Table 4.11, it can be seen in what cases the parameters converge to the right values if the number of iterations is increased. The notation "-" is for non-conducted experiments. (u) and (k) are when the clamping point is unknown and known respectively and (m) is the simulations performed with multiple clamping points with known transformation to an original point. The notation *single* $(dual)$ means that both arms were used but one arm was considered known. Conclusions of convergence in the non-conducted experiments for the single clamped arm could be drawn from other experiments and theory. In the case with known clamping point and identification of the kinematic parameters, the iterations should converge to the correct values since this is very similar to measuring the TCP and hence almost the same as the case of having one of the arms calibrated. For the unknown point, the kinematic parameters will not converge since the point is most likely not the actual point and hence the parameters will not be correct. When identifying both kinematic parameters and offsets for the known clamping point case, the parameters will not converge using the implementation presented in this thesis since it can not separate the offsets from the kinematic parameters.

Table 4.11 Methods where the correct parameters are identified in simulations.

Method	Offsets	Kin.par	Kin.par + offsets
Single clamped (u)	no	-	-
Single clamped (k)	yes	-	-
Single clamped (m)	no	-	-
Single (dual)	yes	yes	no
Dual	yes	no	no

4.2 Experiment results

In this section, the calibration methods are evaluated on real data.

Frida one hand clamping

Offsets In Figures 4.8 the result from the one arm left clamping with all joints offsets and the clamping point set to unknown is presented. In Figure 4.9 the result from the right arm can be seen, and in Table 4.12 the identified joint offsets for both arms can be seen.

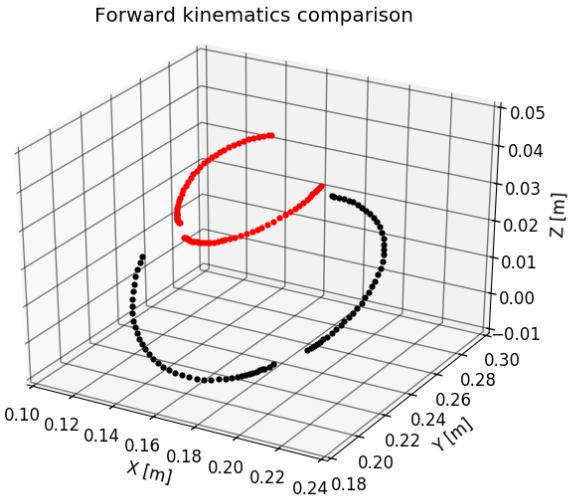

Figure 4.8 Joint offset calibration with real data, left arm. The uncalibrated data points are presented in black, the calibrated in red.

Table 4.12 Offset identification for real data with one arm clamping for 10(3)L and 10(10)R iterations.

Joint	Identified offsets left	Identified offsets right
1	0.01535218	−0.00526062
2	−0.01944153	−0.0219391
3	−0.10272261	0.06114016
4	−0.18232597	0.06929014
5	0.12745788	−0.07205064
6	0.00113391	−0.02431022
7	−0.00246351	−0.01362234

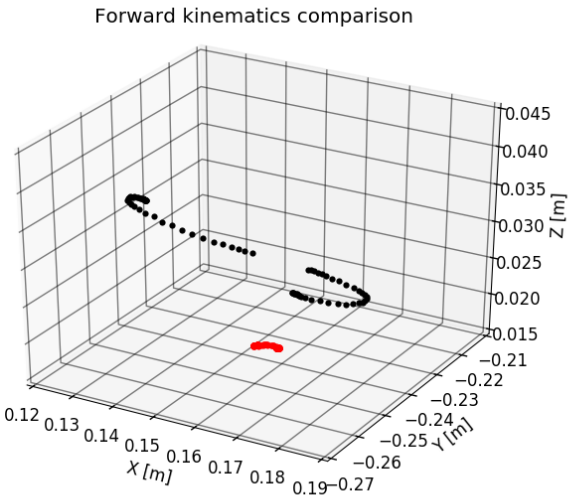

Figure 4.9 Joint offset calibration with real data right arm. The uncalibrated data points are presented in black, the calibrated in red.

Dual arm calibration

Only offset In Table 4.13 the result from the offset identification with the real data is presented. The relative error convergence can be seen in Figures 4.10 and 4.11 (blue bars).

Table 4.13 Joint offset identification for real data with dual arm for 15(7)iterations.

Joint	Identified values Left	Identified values Right
1	−0.02022305	0.02792938
2	0.01346539	0.0534484
3	−0.03011492	0.04263252
4	0.00865558	0.00785042
5	0.00377689	−0.05009817
6	−0.02160491	0.06770496
7	0.03510456	−0.01055271

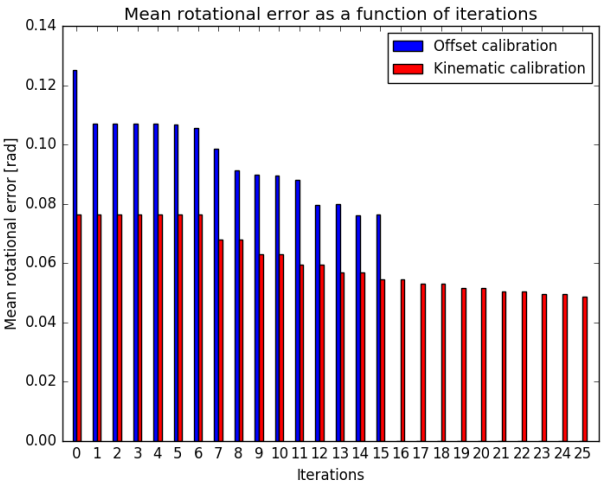

Figure 4.10 How the error in rotation decreases in the offset and kinematic parameters identification algorithm.

First offset, then parameter identification In Figure 4.12 and Figure 4.13, the result from the offset and then kinematic parameter identification is verified on the data from the one hand clamping for the left and right arm, respectively.

In Figures 4.10 and 4.11 it can be seen how the relative errors between the arms change during the course of the iterations in the calibration algorithm. In the algorithm, an identification of the offsets was performed first and the identification of the kinematic parameters was performed after with the updated joint values from the offset identification. Hence, the error at the first iteration in the kinematic calibration starts at the value at the final error of the offset calibration.

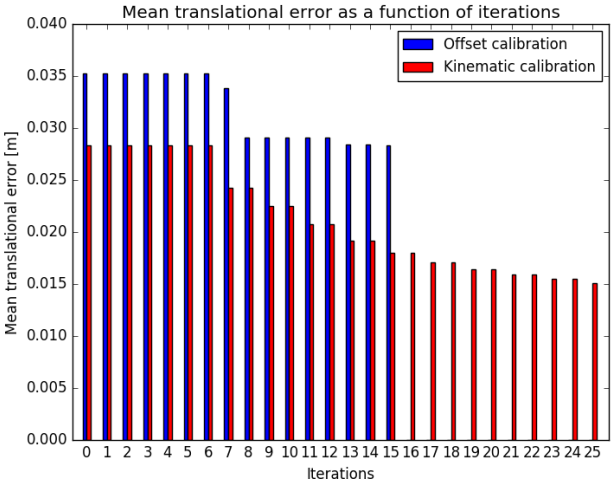

Figure 4.11 How the error in translation decreases in the offset and kinematic parameters identification algorithm.

The error sensitivity of the clamping device identification In Table 4.14 the result from the clamping device analysis in Section 3.4 can be seen. The norm increases drastically when there is an error in the representation of the clamping device. The deviations added represents 1 cm along the x- and y-axes, -1 cm along the z-axis and 0.1 rad positive rotation about y and z and -0.1 rad negative rotation about x. It shows that it is important to get a good estimation of the clamping device.

Table 4.14 Norm if the clamping device is not exactly known for 10(2) and 10(3) iterations.

Norm no error	Norm error
L:0.018233417053	L:0.116347249947
R:0.0094444009505	R:0.115983124183

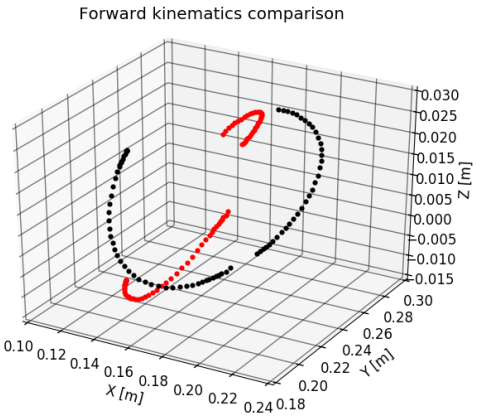

Figure 4.12 First offset identification and then kinematic parameter identification verified on the data from one hand clamping for the left arm. The uncalibrated data are presented in black, the calibrated in red.

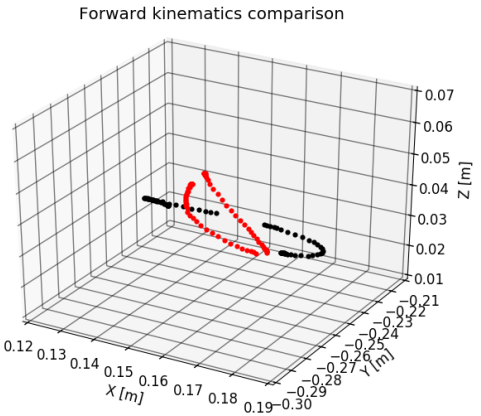

Figure 4.13 First offset identification and then kinematic parameters identification verified on the data from one hand clamping for the right arm. The uncalibrated data are presented in black and the calibrated in red.

5

Discussion and conclusions

The calibration methods evaluated in this thesis are in theory good. They are cheap, easy, and have potential to be accurate. In simulations, it works well for offset identification for one arm with known clamping point, offset identification for dual arm, and kinematic parameter identification for dual arm but with one arm known.

The dual arm offset identification is highly relevant and the clamping device between the two TCPs can be constructed with high precision, which is required. Finally, the kinematic parameter identification for dual arm, but with one known arm is also relevant. If one arm has to be replaced or repaired on YuMi and needs recalibration, the other arm can be used as measurement to calibrate the replaced or repaired arm.

5.1 Simulation result discussion

General error sources for dual arm calibration

An explanation of the poor parameter convergence in some simulations, is the implementation of the algorithm. The algorithm described in Section 2.9 depends on that the pose is known for each arm. In the dual arm implementation, the arms are not constrained to a single point but moves around in space with only the other arm as measurement, where the other arm might be uncalibrated. The errors calculated from (2.55) are least-squares minimizations from all those points. Since simulations (and real experiments) are done with arms that have errors in the kinematic parameters and have joint offsets, the point obtained for each configuration is in fact not the actual point, since an uncalibrated arm is used as measurement. This is an explanation of the poor convergence of the algorithm which can be seen if comparing how well the algorithm converges when assuming one arm to be known as in Table 4.8 compared to Table 4.9, where both arms' parameters are unknown.

In addition to what is discussed in the previous paragraph, the transformation matrix for the clamping device is calculated from a mean of the data, which is also

retrieved with uncalibrated parameters and offsets, which also most certainly gives an uncertainty in the transformation and therefore uncertainties in the identification.

Frida one hand clamping

Offset identification for one point If the clamping point is known, the joint values converge very fast to the correct values, as seen in Table 4.1. Figure 4.1 also shows that the calibrated data have converged to the right location.

In Table 4.2 it can be seen that the joint offsets do not converge exactly to the correct values, when the point is unknown. However, the repeatability is improved as seen in Figure 4.2, where it is shown that the points are much more gathered than before the calibration. The error in the joint offset identification is probably because of the fact that the fixed point is unknown. As mentioned in Section 2.10, the identified point is calculated as a mean of all points, and then the mean is recalculated every iteration with updated joint offsets. This procedure leads to that the identified point might end up in the wrong position, making the joint offsets impossible to identify. Compared to the actual position there is a difference between the true point and the identified one as seen in Figure 4.2.

An alternative to calculate the clamped pose as a mean value would be to set the pose as a free variable in the identification algorithm, and thereby include it in the error model. That, however, that leads to more free variables (for position and orientation), which might complicate the identification. Problems with linear dependent parameters might arise.

The single arm clamping can, as mentioned before, be compared to pressing the palm of the hand towards a table and moving the arm around to different configurations. If one does that, it is easy to notice that the elbow joint does not move at all (or very little). Therefore it is hard to excite the elbow joint (joint three). However, as shown, the identification works well if the point is known, which implies that the excitation is not a problem and that the method with one fixed point works, even though joint three does not move a lot. However, it is important to take this into consideration, because non-excited joints might give a good result in the area where the identification is done, but might give a poor result when the TCP is moved to an other location.

Offset identification for several points A problem with assuming the clamping pose to be unknown is that all joints will not be identifiable. It is hard to identify an offset in joint one with one single fixed unknown point. If there is an offset in joint one, convergence of the offsets in joint two to seven will give the same point relative to the coordinate frame of joint one, just moved in space. It is therefore impossible to distinguish an offset in joint one, from the case when all points are moved in space with unchanged internal distance and orientation. For joint seven it is hard to distinguish a joint offset from a rotation for the fixed point at the same axis. A specific joint offset can either be a joint offset, or a rotation of the fixed point. This constitutes a problem since it is not desirable to require a known orientation

and position of the fixed point. It is hard to get exact measurements and a small measurement error will result in errors in the identification. However, it could be hard to identify the pose by including it in the same error model as parameters and offsets, because of the extra parameters that will be required as mentioned before.

A solution is to use many points mounted with known transformation in between. It is considerably easier to create a plate, or other exact geometry, than to very accurately measure one point's position. As seen in Figure 4.3, Table 4.3 and Table 4.4, this works rather well when the transformation between the points is known. The algorithm does identify the last joint correctly, but the problem with joint one remains. If a geometry is constructed with several clamping positions with known transformation in between, a movement in joint one will just move this geometry in space. The idea is also discussed in [Collin, 2016]. Precisely as above this could be solved by knowing one of the clamping points.

Frida dual arm

Offsets calibration As seen in Table 4.5, the identified offsets almost converge to the actual offsets. Especially joint one and joint seven converge much better than for the one-hand-one-point clamping identification. It is probably due to the fact that the clamping device between the arms has an angle between the attachment points. An offset in joint seven for one arm will have an impact on the configuration for the other arm, and can not be explained by rotating the clamping device. Similar for joint one, an offset in the joint can not be explained by moving the configurations in space, compared to the fixed point case.

As discussed above in the section about offset identification for several points, joints two to seven were successfully identified using several clamping points. Furthermore from Table 4.6 we can see that using the dual arm offset calibration joint one is identifiable for both arms if the other offsets are known, so by combining these two methods a complete calibration of the offsets is possible assuming the kinematic parameters are correct.

Kinematic parameter calibration for one arm assuming the other arm is known As seen in Figure 4.4 and Figure 4.5, the relative error between the arms converges to zero, and remaining errors can be explained by the added measurement noise. In Table 4.8, it is shown that all parameters are identified in the case when only kinematic parameter errors are introduced. With joint offsets added, the relative error between the arms still goes to zero. However, there are small differences between the actual and identified parameters, caused by the fact that the kinematic parameters now also have to compensate for the joint offsets. All parameters were, as expected, identified according to Section 2.9 when only kinematic parameter errors were introduced.

Kinematic parameter calibration for both arms In Figure 4.6 and Figure 4.7 it can be seen that the relative error between the arms is decreasing. However, by looking at the identified parameters and comparing them with the actual parameters,

see Table 4.9, it is obvious that the correct parameters have not been identified. When doing the identification in the dual arm case, the error retrieved between the two arms individually generate a Jacobian for each arm, but when updating the parameters according to (2.56), only half of the parameter updates calculated are applied to each arm in difference to when only identifying the parameters for one arm. This is because the arms are opposite to each other and should converge to a common point, where the translation and orientation differences between the arms are minimized. Adding the whole update may lead to the arms passing each other. This leads to that an increased number of iterations will be needed to be able to converge to a zero error. It also incorporates the above mentioned problem that the arms are not calibrated to start with and that the measurements used as ”actual” position are not correct.

Update the parameters or not

In the implementation of the identification algorithm, there is a criterion that must be fulfilled if the parameters are to be updated. Either, the error must be reduced, or in some implementations, be less than 1.5 times the previous error. As mentioned in Section 2.10, the algorithm sometimes finds a better approach after a step in the wrong direction. There is a restriction on when the parameters are updated, because if the error vector becomes too large, it can result in a too large step in some direction, making the algorithm unstable.

The 1.5 factor (instead of strictly 1.0) was used in the dual arm joint offset simulation (Section 4.1). As seen in Table 4.5 and Table 4.7, the 1.5 factor improved the result. It leads to that the parameters were changed in all 10 iterations, instead of just 2 out of 10. The change to factor 1.5 was useful in the cases where the correct kinematic parameters or joint offsets were found after many iterations as seen in Table 4.11. The change to 1.5 led to a faster convergence after a few iterations. In the cases when the parameters were not fully converged after many iterations, the 1.5 factor did not improve the convergence.

Drawback of simulating in Python

A drawback to just randomly generate configurations with the Python implementation is that it takes no consideration to physical limitations between the robot and its arms (or between the two arms). For the real robot, a generated configuration might be impossible because of collisions between the arms, collisions between the arms and the robot body, or limitations on the maximal joint movements. It might also be impossible to move between two configurations without first detaching the two arms from each other, or detach the arm from the fixed clamping point, reorient one or several joints and then attach the arm/arms again.

5.2 Experiment discussion

In this section, the results from the real experiments are discussed.

General error sources

It was late during the thesis discovered that the last joint on the left arm is broken, and might have been broken when the data sets were acquired. The broken joint has probably introduced a large backlash in that joint and therefore the identified parameters for the left arm are not to be trusted for the one hand clamping case. In the dual arm case, it is highly likely that the broken joint affected the result and is therefore a large error source for the result. However, a benefit with the identification in this thesis is that the broken joint would have been identified long before it was any visible damage on the joint, if the identification algorithm was fully functional from start. The damaged joint would lead to that the identification would fail or give strange results, because of the relative huge backlash in the damaged joint.

In all experiments, all joints, links and tool changers have been considered to be stiff. That is of course not true, and joint and link flexing as well as non-stiff tool changers is probably also an error source.

One hand clamping offset

As seen in Figure 4.9, the repeatability is better after the calibration, but not as good as in simulation. The problem might lie in the lack of excitation of the data. Because of the position of the arm, some joints moved very little and that might explain the circle-formed points in Figure 4.9, compared to the more evenly distributed data points in the simulation presented Figure 4.2.

It is also possible that there are errors in the kinematic parameters, that are not coped with in the offset calibration. There is also a problem with only using one fixed point, as discussed in the section about offset identification for several points.

In Figure 4.8, one can see that the repeatability for the left arm is not at all improved as much as for the right arm, see Figure 4.9. This is most likely a result of the broken joint on the left arm, as discussed in the section on general error sources previously.

Dual arm offset

There is a large difference between the identified offset for the one hand clamping presented in Table 4.12 and the identified joint offsets for the dual arm experiment presented in Table 4.13. None of the algorithms converge as well as in simulation, and there is therefore reason to believe that the identified offsets are not to be trusted. It is likely that the identification of the clamping device's transformation matrix was too inaccurate and that uncertainty makes the algorithm work poorly. Also in this experiment, it is possible that there are kinematic errors that are not coped with.

The broken joint in the left arm is probably also an explanation of the poor result.

Both offsets and kinematic parameters

As can be seen in Figure 4.10 and Figure 4.11, the errors seem to be roughly decreased with 50 percent. However, these are the errors between the two TCPs (including the estimated clamping device). When the nominal parameters were exchanged with the calibrated parameters and the data from the left and right arm clamping were evaluated with the new parameters and with compensated offsets, there was no remarkable change in the errors, as can be seen in Figure 4.12 and Figure 4.13.

5.3 Conclusion

In this thesis, calibration methods for a seven DOF dual arm robot have been evaluated. The calibration methods are easy to use, cheap, and have potential. The algorithms look promising in simulation, has not given the corresponding satisfying result in real experiments. It is hard to tell if it is because of the broken joint seven in the left arm, or not. There are some problems with how the error is handled in the algorithm and the implementation of the algorithm that is discussed in Section 5.1 as well as in Section 5.4.

In simulations it was possible to verify that a total calibration of the offsets is possible by combining the one arm, multiple point clamping calibration and the dual arm offset calibration.

The calibration method did also improve the repeatability in all simulations. It did find the exact parameters in some simulations (see Table 4.11), and the calibration therefore also improved the accuracy. If the error handling problem is coped with, the calibration methods would probably be very useful.

5.4 Further work

To get this algorithm to actually make a significant position and accuracy improvement, there are a few things that can be investigated.

Get a better approximation for the common point in the dual arm implementation

As mentioned in the discussion in Section 5.1, there is probably a problem with how the dual arm algorithm compensates the error between the two TCPs. The compensation has to be further investigated, and maybe changed.

Better identification of the clamping device

An error source is the uncertainties in the clamping device's transformation matrix. If the identification method will be commercialized, it is reasonable to assume that

the clamping device can be designed with high accuracy and that the attachment orientation to each arm will be well known. That will probably improve the result.

Find a way to distinguish identified offsets from identified kinematic parameters

It might be interesting to distinguish the joint offset from the kinematic parameter errors. In this thesis, it is not possible to do that. An offset might be compensated with a change in the kinematic parameters, and vice versa. An improvement of the algorithm would be to find a way to do this distinction.

Elastic joints and links

In this thesis, all joint and links have been considered to be stiff. This assumption means that no consideration has been taken to possible flexing in joints and links. It is hard to construct a model for all flex effects, but it will most likely improve the accuracy of the calibration.

Jacobian identification

It is most likely that joint offsets and ξ-parameters are not identifiable when they are involved in the same error model, according to [He et al., 2010] and also the Jacobian analysis (see Table 4.10). It would have been advantageous if the offsets and parameter errors could be identifiable in the same error model, since modeling and identifying them separately will lead to that the parameters that are identified will compensate for those who are not.

As discussed in Section 2.9, some parameters are probably redundant or non-identifiable. If those parameters would to be identified, it might be possible to remove these from the error model, identify them in a different experiment or assume that they are equal to the nominal value. This procedure will probably lead to a better result. Which parameters to lock have to be rigorously motivated. The solution will probably be to identify the different Jacobians, and thereby find out which parameters that are non-identifiable and/or redundant.

Add restrains in Python simulation

As mentioned in Section 5.1, in the Python simulations in this thesis, there is no control if the generated configurations are valid or not. To generate realistic data, there can not be any invalid configurations, so restrains or a validity check has to be implemented. This check is already implemented in RobotStudio, so it might be possible to use RobotStudio to do this validation, or even generate the data.

A

Appendices

A.1 Kinematic representation

In Table A.1, the mathematical representation of the kinematics for Frida using POE that was used during the thesis can be seen. Each row in the screw matrix represents a screw for that corresponding joint in the robot. $\mathbf{T}_{st}$ is, as earlier mentioned, the initial transformation for the robot which was when all joint angles were zero. The last two matrices are the transformation between the robot base and the right and left arm base, respectively.

A.2 YuMi, IRB 14000

YuMi is a dual-handed robot developed by ABB. It is designed to do small part manufacturing as well as be a collaborative robot, meaning that it requires no safe zones or cages. The safety is inherent in the robot. A model of YuMi is available in RobotStudio, but it has no ExtCtrl implementation. It uses the conventional joint numbering from 1–7, beginning from the base. It can be seen in Figure A.1 [*YuMi* 2016] [*YuMi Overview* 2016].

Table A.1 Mathematical model of the robot kinematics using POE.

Screws	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ -0.030 & 0 & 0.110 & 0 & 1 & 0 \\ 0 & 0 & 0.2565 & 0 & 0 & 1 \\ 0.0405 & 0 & 0.3565 & 0 & 1 & 0 \\ 0.180 & 0 & 0.397 & 1 & 0 & 0 \\ 0.3055 & 0 & 0.3835 & 0 & 1 & 0 \\ 0.3375 & 0 & 0.4105 & 1 & 0 & 0 \end{bmatrix}$
$\mathbf{T}_{st}$	$\begin{bmatrix} 0 & 0 & 1 & 0.3375 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0.4105 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Base transformation (right)	$\begin{bmatrix} 0.5723 & -0.1041 & 0.8134 & 0.0393 \\ 0.6161 & 0.7092 & -0.3427 & -0.0645 \\ -0.5412 & 0.6973 & 0.4700 & 0.4035 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$
Base transformation (left)	$\begin{bmatrix} 0.5784 & 0.1049 & 0.8090 & 0.0423 \\ -0.6143 & 0.7084 & 0.3474 & 0.0626 \\ -0.5367 & -0.6979 & 0.4742 & 0.4049 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$

Figure A.1 YuMi IRB 14000 [*YuMi Overview* 2016].

Bibliography

ABB. *Rapid robot programming language*. URL: http://www.abb.com/blog/gad00540/1DDE6.aspx?tag=RAPID%20programming (visited on 05/17/2016).

ABB (2010a). *RAPID Reference Manual RAPID Overview On-line*. 3HAC 0966-50.

ABB (2010b). *Technical reference manual rapid instructions, functions and data types*. URL: https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf (visited on 06/08/2016).

ABB (2011). *Absolute accuracy, industrial robot option*. URL: https://library.e.abb.com/public/e69d8dd25cd7d36bc125794400374679/AbsAccPR10072EN_R6.pdf (visited on 06/08/2016).

ABB (2014). *Application manual MultiMove*. 3HAC021272-001.

Bagge, F. (2016). *Robotlib.jl*. URL: https://github.com/baggepinnen/Robotlib.jl (visited on 04/29/2016).

BFGS algorithm (2013). URL: https://www.rtmath.net/help/html/9ba786fe-9b40-47fb-a64c-c3a492812581.htm (visited on 07/27/2016).

Blomdell, A., I. Dressler, K. Nilsson, and A. Robertsson (2010). "Flexible application development and high-performance motion control based on external sensing and reconfiguration of abb industrial robot controllers". In: *IEEE International Conference on Robotics and Automation, 2010*.

Chen-Gang, Li-Tong, Chu-Ming, J.-Q. Xuan, and S.-H. Xu (2014). "Review on kinematics calibration technology of serial robots". *International Journal of Precision Engineering and Manufacturing* **15**:8, pp. 1759–1774. ISSN: 2005-4602. DOI: 10.1007/s12541-014-0528-1. URL: http://dx.doi.org/10.1007/s12541-014-0528-1 (visited on 06/08/2016).

Chiaverini, S., G. Oriolo, and I. D. Walker (2008). "Kinematically redundant manipulators". In: *Springer Handbook of Robotics*, pp. 245–268. DOI: 10.1007/978-3-540-30301-5_12. URL: http://dx.doi.org/10.1007/978-3-540-30301-5_12 (visited on 06/08/2016).

Collin, S. (2016). *Kinematic robot calibration using a double ball-bar*. eng. Student Paper, ISSN: 0280-5316.

He, R., Y. Zhao, S. Yang, and S. Yang (2010). "Kinematic-parameter identification for serial-robot calibration based on POE formula". *IEEE Transactions on Robotics* **26**:3, pp. 411–423. ISSN: 1552-3098. DOI: 10.1109/TRO.2010.2047529.

Ionescu, H. (2010). URL: https://commons.wikimedia.org/wiki/File:6DOF_en.jpg (visited on 06/01/2016).

Kock, S., T. Vittor, B. Matthias, H. Jerregard, M. Källman, I. Lundberg, R. Mellander, and M. Hedelind (2011). "Robot concept for scalable, flexible assembly automation: a technology study on a harmless dual-armed robot". In: *Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on*. Tampere, Finland, pp. 1–5. DOI: 10.1109/ISAM.2011.5942358.

Lecture 8, Backlash and Quantization (2015). URL: http://www.control.lth.se/media/Education/EngineeringProgram/FRTN05/2015/lec08_2015.pdf (visited on 06/08/2016).

Linear Least Squares Problems (2013). URL: https://www.math.washington.edu/~morrow/498_13/demmelsvd.pdf (visited on 04/24/2016).

MATLAB (2016). URL: http://se.mathworks.com/products/matlab/ (visited on 07/14/2016).

MIT (2011). *Singular value decomposition*. URL: https://ocw.mit.edu/courses/mathematics/18-06sc-linear-algebra-fall-2011/positive-definite-matrices-and-applications/singular-value-decomposition/MIT18_06SCF11_Ses3.5sum.pdf (visited on 10/02/2016).

Murray, R. M., S. S. Sastry, and Z. Li (1994). *A Mathematical Introduction to Robotic Manipulation*. 1st. CRC Press, Inc., Boca Raton, FL, USA. ISBN: 0849379814.

Nof, S. (1999). *Handbook of Industrial Robotics*. Electrical and electronic engineering v. 1. Wiley. ISBN: 9780471177838. URL: https://books.google.se/books?id=7od4alFKfNMC.

NumPy (2016). URL: http://www.numpy.org/ (visited on 06/01/2016).

Okamura, K. and F. Park (1996). "Kinematic calibration using the product of exponentials formula". *Robotica* **14** (04), pp. 415–421. ISSN: 1469-8668. DOI: 10.1017/S0263574700019810. URL: http://journals.cambridge.org/article_S0263574700019810 (visited on 06/08/2016).

Park, F. C. and K. Okamura (1994). "Advances in robot kinematics and computational geometry". In: Lenarčič, J. et al. (Eds.). Springer Netherlands, Dordrecht. Chap. Kinematic Calibration and the Product of Exponentials Formula, pp. 119–128. ISBN: 978-94-015-8348-0. DOI: 10.1007/978-94-015-8348-0_12. URL: http://dx.doi.org/10.1007/978-94-015-8348-0_12 (visited on 06/08/2016).

RobotStudio (2016). URL: http://new.abb.com/products/robotics/sv/robotstudio (visited on 05/17/2016).

SciPy (2015). URL: http://docs.scipy.org/doc/scipy-0.15.1/reference/index.html (visited on 06/01/2016).

Simulink (2016). URL: http://se.mathworks.com/products/simulink/ (visited on 07/14/2016).

Spong, M. W., S. Hutchinson, and M. Vidyasagar (2006). *Robot modeling and control*. John Wiley & Sons, Hoboken (N.J.) ISBN: 0-471-64990-2.

Stone, H. (1987). *Kinematic Modeling, Identification, and Control of Robotic Manipulators*. The Springer International Series in Engineering and Computer Science. Springer US. ISBN: 9780898382372. URL: https://books.google.se/books?id=6ZZhjW3EWugC (visited on 06/08/2016).

SymPy (2016). URL: http://www.sympy.org/en/index.html (visited on 06/01/2016).

VirtualController. URL: http://www.abb.com/product/seitp327/30450ba8a4430bcfc125727d004987be.aspx (visited on 05/17/2016).

Yang, X., L. Wu, J. Li, and K. Chen (2014). "A minimal kinematic model for serial robot calibration using POE formula". *Robotics and Computer-Integrated Manufacturing* **30**:3, pp. 326 –334. ISSN: 0736-5845. DOI: http://dx.doi.org/10.1016/j.rcim.2013.11.002. URL: http://www.sciencedirect.com/science/article/pii/S0736584513000999 (visited on 07/27/2016).

YuMi (2016). URL: http://new.abb.com/products/robotics/yumi (visited on 06/01/2016).

YuMi Overview (2016). URL: https://library.e.abb.com/public/4bcf2603f76f49088b80f7f1c49045eb/IRB14000ExternalVersionFinal.pdf (visited on 06/01/2016).

Zhuang, H., L. K. Wang, and Z. S. Roth (1993). "Error-model-based robot calibration using a modified CPC model". *Robotics and Computer-Integrated Manufacturing* **10**:4, pp. 287 –299. ISSN: 0736-5845. DOI: http://dx.doi.org/10.1016/0736-5845(93)90042-I. URL: http://www.sciencedirect.com/science/article/pii/073658459390042I (visited on 06/08/2016).

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

Document name
MASTER´S THESIS

Date of issue
December 2016

Document Number
ISRN LUTFD2/TFRT--6012--SE

Author(s)
Gustaf Bergström
Björn Green

Supervisor
Olof Sörnmo, Cognibotics AB
Andreas Stolt, Cognibotics AB
Björn Olofsson, Dept. of Automatic Control, Lund University, Sweden
Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Sponsoring organization

Title and subtitle
Evaluation of Calibration Method for Redundant Dual Arm Industrial Robots Using Internal Sensors

Abstract
To maintain a high position accuracy in robots is essential in today's industry. Because of different factors, as for example wear and tear, the robots may need to be recalibrated after a while to maintain their high precision. Today the calibration can be costly. The robot is often sent away to be calibrated and will therefore cause stoppages in production, and if the calibration is performed in-house, education of staff and purchases of expensive external measurement equipment are required. In this thesis, a calibration method for the dual arm, seven degrees of freedom (per arm) robot YuMi [YuMi 2016] from ABB is evaluated. The idea is to connect the two arms to each other, perform specified movements and use solely the internal sensors in the robot to do the calibration. The arms will also be docked in fixed positions in the robot's work space and while fixed, they will be moved into different configurations to obtain additional measurement data. A challenging part is that the robot has an extra joint in each arm compared to common six-axis industrial robots, which makes the robot inherently redundant. The redundancy makes it possible to reach a certain point with the robot tool using several different arm configurations, this makes it more difficult to estimate the robot's parameters.

In this thesis, several different types of calibrations were performed. Because the calibration methods have some drawbacks that need to be further investigated, the correct parameters were only identified in some cases. In those cases where the real parameters were not found, only the repeatability, and not the accuracy, was improved. The methods showed these results in simulations, while the actual robot was not successfully calibrated in experiments.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-79

Recipient's notes

Security classification