CHALMERS

Planning, Programming and Control of Dual-Arm Robot Contact Operations

MASTER'S THESIS IN THE MASTER DEGREE PROGRAMME, SYSTEMS, CONTROL AND MECHATRONICS

Daniel Andersson

Product and production development
Division of Production Systems
Chalmers University of Technology
Gothenburg, Sweden 2011
Master Thesis 2011

Abstract

Recent industrial development of dual-arm robots has considerably leveraged interest of researchers and robot manufacturer. The big difference for this concept compared with normal single arm robots, is that these robots can save space and are able to do multi-tasking. The way human plan and control bimanual operation was investigated and it was learned that there was much that could be learned from human motion planning. A dual-arm robot from pi4_robotics which is the same size as a human has been used. One bimanual motion operation, where two arms are holding a beam has been analyzed in this thesis. In order to not damage the robot, a simulation model of the robot was made in Simulink. It is desired that no internal forces or moments is applied to the beam and that such forces would be compensated when the beam is gripped. To ensure a safe system, the simulation functions has been programmed in C++ so that it is possible to try these in the simulation environment before they are used by the programming environment of the robot.

In this thesis several difference dual-arm control methods for controlling two arms simultaneously have been investigated. The benefit and disadvantages between the difference methods has been pointed out and finally an impedance controller with position mode was selected to control the motion of the arms. The main advantage with this method is that it can control the behavior of the beam and does not need to switch between different kind of controllers. The impedance control algorithm was implemented and motions were the grippers compressed a beam was simulated. During the simulations special attention has been put on reducing the internal forces of the external object while solving the desired motion task.

An equation to calculate the internal forces of the beam was derived and the estimated force was given to an impedance controller. The results show a promising response of the controller where internal force is approaching zero without having any undershot. It takes about half second for the internal forces to be compensated. The algorithm works well when the robot is moving in the free space as well as during the contact phase.

Keywords: dual-arm, Robot programming, Robot assembly, Bimanual operation, Impedance Control, Simulink

Acknowledgements

This Master's thesis has been elaborated at Fraunhofer Institute IPK in Berlin, Germany. Many people have supported me to accomplish this thesis and writing this report. I would first like to thank my supervisor Ph. D Dragoljub T. Surdilovic at Production Systems of Fraunhofer-Gesellschaft and my supervisor Rolf Berlin at Product and Production development department of Chalmers university of technology. I would also like to thank Prof. Dr.-Ing. Jörg Krüger at Fraunhofer-Gesellschaftfor supporting the project. Moreover I would futhermore like to thank Prof. Johan Stahre at Chalmers Product and Production development, Ph. D Johan Carlson at Fraunhofer-Chalmers and Matthias Krinke at Pi4 for showing interested in the project and offering their help in case of any questions. Many thanks also to Johanna Eckardt, Alexander Börjesson and Adam Svensson for reviewing my report and to my colleagues at Fraunhofer Pham Xuan Ba, Axel Vick, Nikolay Bogdanov as well as the previous colleagues at Fraunhofer IPK which have been working on the project.

Daniel Andersson, Gothenburg 1/6/2011

List of Abbreviations

CP	Continuous Path
DOF	Degree of Freedom
FRIDA	Friendly Robot for Industrial Dual-arm Assembly
IMCO	Impedance Controller Module
IPK	Institute for Production Systems and Design Technology
IPO	Interpolation Module
MABA - MABA	Men Are Better At - Machines Are Better At
MEX	MATLAB Executable
MinGW	Minimalist GNU for Windows
MPC	Model Predictive Control
PID	Proportional Integral Derivative
PTP	Point to Point
RCC	Remote Center of Compliance
SDA10D	Slim, Dual-Arm Robot with capacity of 10kg payload per arm
TCP	Tool Center Point

Contents

1

Introduction

Dual-arm robot is a robot with two arms which which are meant to be capable to perform flexible operations in assembly lines of Small and Medium Enterprises, commonly performed by humans. The overall goal of the concept is to keep human workers in the loop and to support them with powerful tools. It should be emphasised that humans are the most flexible element of an assembly line, offering many advantages over robotics. Rather than replacing a human, a robot could be integrated and support humans with qualified tools or to work in hazardous environments. This combines the human creativity, intelligence and skills with the advantages of technical system with better physical power, speed and accuracy. A picture of a dual-arm robot from pi4_robotics which this thesis has focused on is given below in Figure 1.

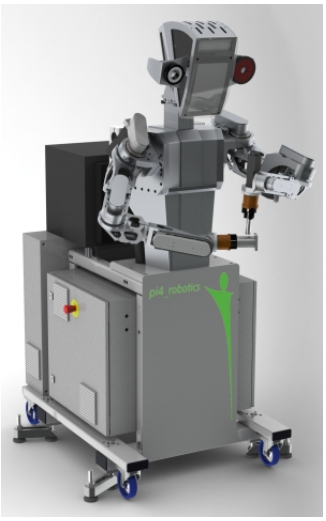

Figure 1: Picture of pi4_robotics industrial robot pi4_workerbot [1].

The main advantages and benefits of the dual-arm robot concept over single arm systems are: multitasking and space saving [2]. The disadvantages are higher cost as well as that the technology is not yet well developed. The concept is therefore being researched and today there are many companies which believes in the idea of having dual-arm robots in the industry. Other than pi4_robotics, Motorman has released a dual-arm robot called SDA10D assembly robot [3], KUKA/DLR has built a humanoid robot for international space station called Justin [4] and ABB has also recently presented their concept robot FRIDA [5]. In common they all have that they have developed robots which will operate in human-occupied environments. Also other companies contributes to the development of new robotic systems by doing tools to the robots, for example by making better and innovative grippers, sensors and more intelligent surveillance systems.

Robotic components are becoming cheaper, computer power is increasing and there will become a big need of service oriented task in a couple of years [6]. There are several areas robotics can be improved in order to work well close to humans and to be capable to perform all the manual tasks. For the different research areas, the desired goals can be compared with the knowledge of a child[6]. Researchers want to achieve object recognition good as a 2-years-old-child, for example to be able to recognise if people are young/old or male/female. The language understanding should be as good as a 4-year-old child, which mean it should be able to understand different accents or during influence of much noise. The manual dexterity, or the sleight of the grippers should be as good as a 6-years-old child, for example to perform simple assembly instructions. Finally the social understanding of a robot should be equivalent with a 9 years old child. This mean that the robot is able to understand what other people around themselves see and understand [6].

The field of motion planning and assembly operations will be highlighted in this thesis. There are, however, several practical difficulties in understanding, controlling, planning and programming the motion for dual-arm robots. A practical approach to dual-arm motion planning is based on mimicking human operations since most of the desired tasks are defined for humans. The bimanual operations performed by humans are mostly based on relatively simple motions in both arms, commonly symmetric and can work dependently or independently of each other. It was recently recognised that the main reason for simplifying arms motion is due to the very complex planning of the two arms in real time. Humans generate simple trajectories which are easy to synchronize and monitor during the execution [7].

The main control problems in dual-arm robots are related to the physical contact and interaction between arms (constrained motion) and environment in bimanual contact operations. Impedance control provides a common control approach to cope with uncertainties in robotic arms and environment, as well as to maintain interaction forces within some desired level. Controlling and programming of dual-arm assembly tasks, require however more complex approaches. The assembly tasks involve various composite

motion and transition phases, e.g. from free-space motion, via unilateral force contact towards completely constrained closed-chain motion of coupled parts. Several researches address the multi-arm motion planning problems focusing common object path planning, motion coordination, collision avoidance etc. Only a few investigations focused on bi-manual compliance control and assembly process planning.

1.1 Background and Purpose

During the last decades a lot of automatisation has been done in industry, especially in the developed countries [8]. In several areas industrial robots have been implemented in the assembly lines in order to increase the efficiency and quality, as well as to reduce the production cost. The question of what to automate has no simple answer. The "Men Are Better At - Machines Are Better At" (MABA-MABA) lists have tried to find a simple answer, however such work relies on a presumption of fixed human and machine strength [9]. Instead, the more promising tactic today is how to get human and automation get along together. Several projects have been developed to improve the cooperation between people and machines. As in sport, good team players make their activities observable for their fellow team players. This could also be applied to a working team. It is important to have a good communication and understanding between the different workers in the team whether the workers are machines or humans [10]. The cooperation between human with automation and machines has to be thought of from the design phase, and exist in the manufacturing, installation, operation, as well as in the maintenance phases [8].

There is a wish among most automated industries to optimise production even further in order to increase efficiency and to be better than their competitors. The complexity of the robotic tasks is therefore getting more and more advanced, which has introduced the customers need of cooperation between several arms in order to solve the objectives [11]. In general, today this is solved by having several robots next to the line which are being controlled by an external control unit. Several arms are capable of carrying heavier load and can perform more complicated task than the capacity of single arm robots [11]. The idea of switching from a single arm robot to a robot with several arms is a natural approach considering how the biological evolution has been developed. The main advantages and benefits of dual-arm robots over single arm systems are multitasking and space saving [2]. That the environment around the arms are good defined will also make a robot platform easier to install, move around and to calibrate. These are all valuable benefits which could economically motivate development of dual-arm robots. Given a standard environment of the arm, it is possible as well to design a more user friendly development platform which will make it much easier for an operator to program a bimuanual operation. For example, the user interface could be designed to move the object instead of directly controlling the arms.

The main purpose of this thesis work is to contribute a better understanding of

bimanual robot contact instructions including analysis of transition phases and specific motions. The aim is to generate a platform of bimanual task primitives and to decompose complex tasks into dual-arm primitives and skills. The work of previous research will be investigated for the implementation of a dual-arm simulation model. In the beginning of the thesis only simulations of single arm operations had been tested. The thesis will afterward be the base of further development at Fraunhofer-IPK of more advanced dual-arm algorithms that can be designed for a dual-arm robot.

1.2 Project questions

How should an algorithm be constructed in order implement a robust and safe dual-arm motion planning that would be practical in the industry? To answer that question, the following sub questions have been created:

- Why would the industry be interested of a dual-arm robot?
  - Are there many robotic companies that have developed dual-arm robots?
  - Whats the big advantageous of dual-arm robot compared with single arm robots?
- What can be learned from Human bimanual control of the arms?
  - How does the brain coordinate both of the arms?
  - What kind of motion is the brain good at and less good at?
  - Does the brain plan the motion far in advance in order to avoid singularities?
- What have already been done at Fraunhofer IPK?
  - How does the current simulation environment work?
  - Which motions operation are of interest for further development?
- How could a dual-arm control algorithm in the best way be implemented?
  - What considerations have to be made before running practical experiments?
  - How can the contact phase be modeled?
  - Do we need a model of the external environment?

1.3 Objective

For each skill the control algorithm should be developed based on robust interaction control and design framework. The specific thesis goals relate:

- Analysing of the human bimanual motion operation.

- Integration of a simulation environment for basic manual contact tasks.
- Extension of a compliance control for bimanual interaction control - considering all transition phases.
- Design and programming of algorithms for basic bimanual assembly skills and tasks.
- Experimental evaluation of the bimanual assembly operation control.

1.4 Limitations

The analyses of human bimanual motion planning will be limited to the area that could be beneficial when a developing dual-arm planning algorithm. The thesis will be focused on the development of the existing platform and control systems which are currently used for the existing platform at Fraunhofer-Gesellschaft in Berlin. One movement operation, when two arms are holding a beam, will be looked into detail (Bi-Hold in Figure 2, section 2.1.1). Modules from previous research will be used to design a sufficient dual-arm system for analysing the motion operation. This so that previous results, such as stability and safety margins could be applied to the dual-arm system. The tasks, which will be studied of the dual-arm robot will be basic operations where maximum one external force is present. Cost and investment aspects of dual-arm robots will not be treated in the report.

1.5 Organisation and timeline

The thesis has been done during a time of 20 working weeks during the spring 2011. The first objective of the thesis was to research and understand the current system architecture of the existing platform. Different control techniques as well as the robot hardware limitations have been studied. Afterwards the knowledge was used to develop a control algorithm for the bimanual contact task. The intention is to implement and test an algorithm on a dual-arm robotic system after this thesis by Fraunhofer-IPK.

The report is organized as the following:

- **Section 2.1** gives an overview of the Bi-manual control.
- **Section 2.2** deals with the contact task and explains the different control techniques which can be used to approach this operation.
- **Section 2.3** explain the theory of internal forces and show how these could be calculated.
- **Section 2.4** presents the geometric model of the robot which the experiments will be performed on.

- **Section 3.1** describes the real-time implementations for the pi4_Workerbot.
- **Section 3.2** shows how the simulation environment is structured and how it is connected with the programming of the actual robot.
- **Chapter 4** presents the results of the simulations.
- **Chapter 5** discusses the implementation and results of the internal force calculation, additional development and dual-arm robotics in general.
- **Chapter 6** ends with conclusions and gives some recommendations for further work.

2

Modeling

The motion operation when two arms are holding a beam will be modeled and investigated. In order to get a good understanding of dual-arm control and planning as well as to find out which control method that will suit this motion operation, several different areas has been investigated. The first section 2.1 describes the bimanual control from a system approach and describes the different bimanual movement operations needed for assembly.

Section 2.2 describes the theory of control methods during contact interaction and the different control algorithms which could be used. Practically problems as well as desired performances has been taken into consideration when comparing the algorithms. The selection of control algorithm is very much dependent of the task and might not be the same for any objective.

In order to get a good performance, stability and robustness during the contact transition, it is important to find a way to estimate the force generated outside the robot. The case where the robot is holding a beam with two hand has be analysed and a way to calculate the internal force of this beam is derived in section 2.3.

As last section in this chapter, section 2.4 shows the model of the analysed robot and defines the coordinated frames as well as how to transform between the frames. This part is handy for implementing the functions in chapter 3.

2.1 Bimanual movements

Bimanual movements means a coordinated movement of two arms in order to solve a desired objective. To approach the controlling of the dual-arm robot the fundamental assembly tasks of the human arms is analysed and then defined as different operations shown in Figure 2 [12]. These has been named Bi-Approach (Retract), Bi-Grasp (Release), Bi-Insert (Extract), Bi-Hinge, Bi-Slide, Bi-Hold, Bi-Yield and Bi-Move. The movement operation Bi-Hold will be investigated in this report.

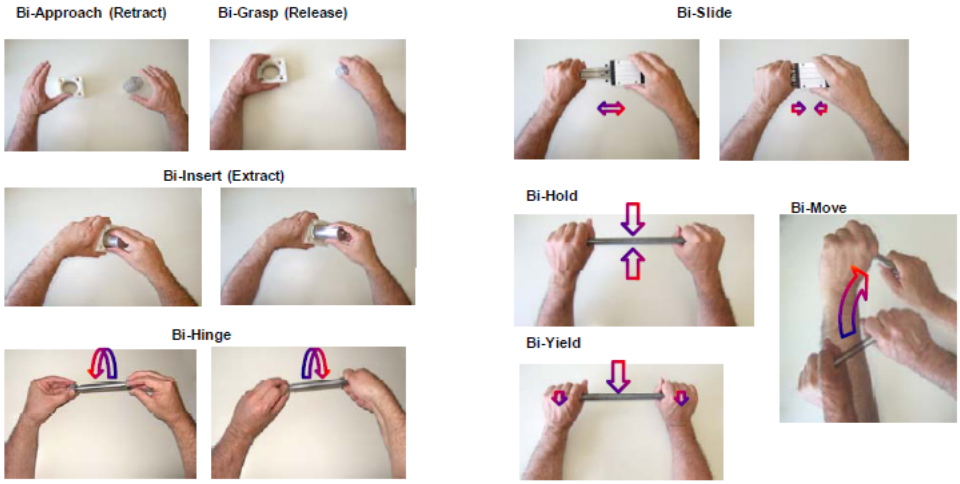

Figure 2: Bimanual movements operations [12].

2.1.1 Human planning

A practical approach to dual-arm planning is based on mimicking human operations. The way the human brain is working to control and plan trajectories of the body's two arms is a very interesting research which has during the last decade made significant progress [7]. The research of the knowledge of what the human actually is capable of increases and it gives an understanding the possible objectives for dual-arm robot. An insight how the brain is working is also being a central interest to neurophysiologists. Functional image techniques (fmRI, PET, MEG) has permitted scientist with many discoveries to understand how the brain is working [7].

It is generally accepted that the movement of each of the arm is controlled in the motor cortex of the brain. The right half of the motor cortex area controls the left side of the body, and vice versa [7]. The fact that each arm is actually being controlled individually contradicts the idea of having a dual-arm control algorithm. However, it turns out that the two sides are very strongly coupled and some theories insist that the supplement motor area has a special role in the path planning including two arms [13]. Image techniques have registered that some neurons are active only during bimanual

movements and not during unimanual movements of either arm, which verifies the coupling idea [7]. Split-brain patients, who are humans missing the ability to communicate between the left and right sides of the brain, have difficulties in coordinating the arms [13]. Interesting is, that they are better than a healthy person to perform very different kind of movements at the same time. When a healthy human tries to draw a circle with one hand and a line with the other, both hands usually end up with making two ovals. It turns out that human only is good to perform symmetric or mirrored movements with both hands simultaneously. Limiting the motions of the two arms into a subset of only symmetric and mirror-symmetric motions are not necessary for programming a robot. A figure from the test with spit brain patients is given in Figure 3.

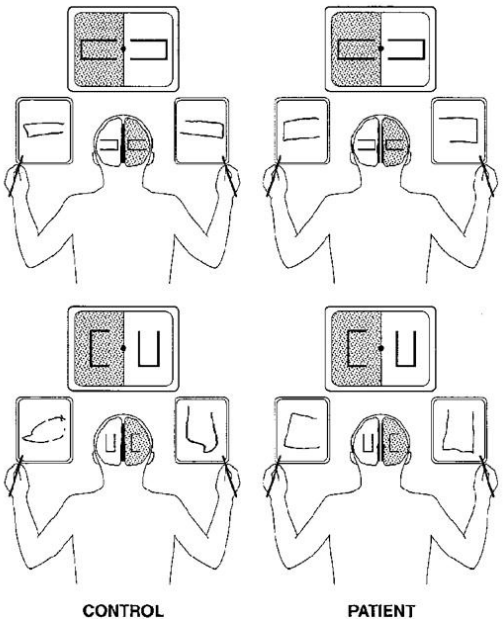

Figure 3: Test made on Split brain patients compared with healthy persons [13]. Permission to be reused by Copyright Clearance Center.

Another interesting topic relates to the motion and how the brain is avoiding singularities. It is still not very clear whether the brain plan the whole or only a part of a motion in order to avoid singularities but it seem like the brain generate simple trajectories which are easy to synchronize and monitor [7]. Another theory is that the brain has a database and has learned from mistakes in early ages. The singularities are path solutions where at least one of the actuators has a reached a solution which is faster than the maximum speed for the actuator. The reason a simple controller could not take care of this, is that such constraints are in fact non linear and solving the path in respect to these increase the computational power significant. There are however non linear methods which could take care of such problems in real time, for example optimal control or MPC (Model Predictive Control) [14]. If the arm will move in the same path as initially when one actuator is close to a singularity, then all the

velocities of all actuators have to be reduced. The area is currently a big challenge in the controlling of the dual-arm robots since it is often desired to find the fastest solution.

2.1.2 Dual-arm planning

Several approaches of different control policies have been made to design Bimanual controllers [2]. These control policies decides how the two arms is being controlled in order to be able to execute bimanual motions. The most basic and the first one to be described is called the "Master/Slave" approach. This means that one arm controlled and that this arm then is coordinating the other arm. There are also indications that human arms have a similar behavior since most people have a dominate arm which usually execute the motions a couple of milliseconds (15-39 ms) before the non-dominate arm [13]. However, the "Master/Slave" approach has no interaction with the environment and is therefore not so attractive for many tasks, such as assembly. It is important that the controller is able to handle uncertainties. The uncertainties could also be different between the different arms. Another approach is called the "Coordinated Motion" approach. In this approach the arms are controlled independently. This design approach also has a problem to handle the uncertainties and also to control the behavior of external objects [15]. For example to move an object from one point to another without applying to much force to the object.

A more promising type of approach is the "Object Motion" policy [15]. In this control policy the behavior of the manipulated object is specified, for example the desired motion. There has been a lot of different control algorithms developed and implemented in this field. To mention a few; object position control, object force/position control and object Impedance control. The control implementations determine the performance of the algorithm.

The motion of a dual-arm is complex and using an object motion strategic control policy would instead specify the external object behavior than the movement of each arm. The behavior of the external object is highly interesting when analysing the motions during contact. A control method which does not manage to compensate for uncertainties during the contact could damage the robot as well as external object significantly. The implementation of the control policy can be done in different ways which also affects the results [15]. In the next section the different ways of implementing the "Object Motion" policy is beeing described.

2.2 Contact operations

Several different methods to solve the control of a dual-arm robot during a contact operation have been evaluated. For a good performance of an implemented control algorithm for contact operations the following criteria have to be fulfilled[2];

- Accurate trajectories
- Compensate inertial forces
- Resolve natural redundancy
- Less computational resources

Achieving very good performances for all of these points is difficult and therefore usually a trade-off between the criterion has to be made. It should directly be noted that during other robot motion operations, the performances are dependent on other factors. For example for movement operations in free space, very accurate trajectories are usually not needed. Most important is that the control algorithm does not jeopardize safety and stability. The different control techniques that has been evaluated can be divided in two big branches, the ones using passive compliance algorithms and the ones using active compliance algorithms. These two fields are explained in the next two subsections.

2.2.1 Passive compliance

The passive compliance control algorithm are algorithms which indirectly solve the problem when there exists an error between the robot and the environment [2]. Some systems solve this in a fixed, non adaptive way. This can for example be done by having some flexibility in the structure or via an external device. A revolutionary mechanical device invented in the 80's, is the RCC, which is a device that corrects the rotation and forces to desired values between the arms and the environment. However, since the RCC is a mechanical device which includes translation and rotation parts it is facing problems with accurate results due to the gravity and angular velocities. The usage of RCC is therefore restricted to some operation ranges.

The other group of passive compliance control algorithms are the ones that have an adaptive performance [2]. These systems can for example have adjustment of the servo gains or additional devices which can adjust the compliance. The adjustment is however not done automatically and need a lot of tuning.

2.2.2 Active compliance

The active compliance control algorithms are algorithms where the force or position is being feed back to the controller [16]. This can be done in several different ways. Most of the active compliance control algorithms can be ordered into two different groups. One is

called the "Position/Force control" or sometimes "Hybrid Control" and the other group is called "Impedance Control". The active compliance algorithms need more computational resources than passive control but the calculations are not that heavy so that it would be a problem to practically use them in real time applications [2].

2.2.3 Position/Force control

The Position/Force control assumes that the controlling of the force and the controlling of the position are non conflicting [16]. The control of the correct force is handled according to the environment and the control of the position is done according to the manipulator [16]. In the case when these controls are conflicting the designer has to decide which of these controls that has the highest importance. For some objectives this could be a very good solution. For example if the objective requires a force of a certain magnitude in order to successfully execute the task. However, the main problem with Position/Force control is that the position controller requires a rather stiff manipulator in order to have an accurate position while the force controller requires a rather elastic manipulator in order to have an accurate force. This is not a problem as long as the position and force requirements are in different directions, but often this is not the case. For example to hold a sensitive object such as an egg with two arms requires both a position & force requirement in the same direction, hence such a task would not be suitable for the Position/Force control.

2.2.4 Impedance Control

Instead of controlling the position, velocity or force, an idea is to enforce the relation [15]. This is the concept of impedance control where the system is being told to act like a damped-spring-mass system. The reason of selecting this simple is that the system then behaves in way which is easy for the operator to understand. The tuning parameters have units that have physical meanings. If the manipulator hits a surface, then the contact force could be predicted by the operator if he knows the parameters. The negative part is that this control require the dynamic of the arms as well as external objects. There are several different kind of impedance control. The general form is given in equation 2.1, but there also exist models with only the stiffness or damping parts of the model [15].

$$F = M_{\triangle}\ddot{x} + \beta_{\triangle}\dot{x} + K_{\triangle}x \tag{2.1}$$

The variable K corresponds to the stiffness of the system. A high stiffness is desired for the case when the accuracy is very important and a small value of K corresponds to that small interaction forces should be compensated. The model of the system can be in several DOF. Usually sex DOF is chosen to correspond to motions and rotations in the x, y, z, roll , pitch and yaw. The β value is a parameter for the damping of the system. A large damping coefficient mean that the system should dissipate much energy. The M coefficient describes the mass of the systems and therefore a good tuning variable to describe the transient behavior during contact. Contact essential operations such as

grinding and polishing should not be controlled by an impedance controller since these operations need a force of a certain magnitude [2].

There are several ways to evalate the system. In general term this is done by looking at the error, given by difference between the model and the real system. Either the motion could be stabilized in respect of the force or the force could be stabilized in respect to the motion. The error is normally defined as e_p when position is used and e_f when the force is used. The verification model for the force is given by equation 2.2 where F in this case is the force measurement [2].

$$e_f = M\ddot{e} + \beta\dot{e} + Ke - F \tag{2.2}$$

A problem with the force based verification model is that many systems in industry is designed as position devices. This mean that the robot are missing an accurate joint-torque control which is needed for the verification. In the case of position based verification there is another problem which is that it is not able to handle soft impedances. A combination between the models could therefore be desired. The system of the position mode with force as outer loop is given in Figure 4.

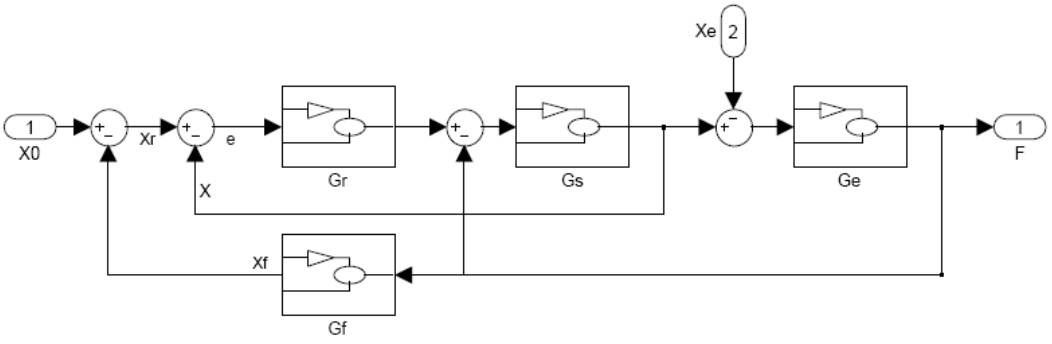

Figure 4: Impedance Controller with outer force loop.

where

- Gf is the target admittance.
- Ge is the environment.
- Gr is the position controller.
- Gs is the plant, in this case the robot.

From the Figure 4 the error e_p can be obtained according to equation 2.3 [2].

$$e_p = Xr - X = X0 - Xf - X = X0 - X - Gf^{-1}F \tag{2.3}$$

Here Xf is the target position deviation, $X0$ the position of the arm and X the actual position. The target admittance Gf is obtained as a solution of the target model difference equation 2.1 [2]. As can be seen the inverse kinematic of the system need to be calculated. This operation have several solutions and the best need much computational power. However it is not essential that the best path is used. In the case of force mode, a similar equation where the inverse of the force need to be calculated. Using position controllers in the joints are much more accurate than force sensors and therefore the position mode is the most practical approach for industry. It should however be noted that biology does not have position sensors use a solution more similar to the force mode. For humans the position is also feedback with the eyes to control the motion. In order to verify the systems in the simulation environment, an external objects will need to be simulated. This is described in the next section.

2.3 Internal forces

This section will describe how to make a function calculating the internal forces as well as moments in a simulation of a beam that is being hold and compressed/expanded by two robotic arms. The purpose of this function is to correct the arms in such a way that too big forces are avoided to the object. However the arms are never allowed to lose contact with the object. As explained in section 2.2 this can be solved in many different ways, with external devices or by control algorithms. An impedance control algorithm is decided to be used for problem to control the internal forces. Essential information is therefore to measure or find out a way to calculate the internal forces which are being created by the robots two arms. It has been assumed that there are no ways to measure the internal forces directly, like having an extra sensor. Instead the force and momentum will be measured by two force sensors located at the wrist of the left and right hand of the robot.

2.3.1 Mathematical derivation

The calculation of the internal forces will later be used to feedback to the controller so that the dual-arm robot will hold the beam with applying minimal internal forces and momentum. The vector of the force, F and moment, M, is defined as W according to (2.4).

$$\begin{bmatrix} F \\ M \end{bmatrix} = W \tag{2.4}$$

Both F and M are here $3x1$ matrices which mean that W has the dimensions $6x1$. It is interesting to know the forces of an arbitrary point C. Forces and moments are mapped from the point of sensor S to the compliance point C according to (2.5).

$$W_T = {}^C J_S^{-T} \cdot W_S \tag{2.5}$$

Here the matrix J (Jacobian) from the sensor frame to compliance frame with the dimensions $6x6$ can be written as (2.6) [2].

$${}^C J_S = \begin{bmatrix} {}^C R_S & -{}^C R_S \cdot \underline{\underline{SC}} \\ 0_{3x3} & {}^C R_S \end{bmatrix} \tag{2.6}$$

where ${}^C R_S$ is a rotation matrix with the dimensions $3x3$ from S to C and

$$\underline{\underline{SC}} = \begin{bmatrix} 0 & z_S - z_C & y_C - y_S \\ z_C - z_C & 0 & x_C - x_S \\ y_S - y_C & x_C - x_S & 0 \end{bmatrix} \tag{2.7}$$

By splitting up the equation (2.5) to the left and right sides sensors it is rewritten to equation (2.8). L stands for the point where the left robot arm holds the beam and R stands for the right side.

$$W_T = {}^C J_{SL}^{-T} \cdot W_{SL} + {}^C J_{SR}^{-T} \cdot W_{SR} \tag{2.8}$$

$$W_T = \begin{bmatrix} {}^C J_{SL}^{-T} & {}^C J_{SR}^{-T} \end{bmatrix} \cdot \begin{bmatrix} W_{SL} \\ W_{SR} \end{bmatrix} \tag{2.9}$$

By naming these two matrices to the right, ${}^C \hat{J}_{SLR}^W$ and the two to the left $\hat{W}_{SLR}$, equation 2.9 can also be written as

$$W_T = {}^C \hat{J}_{SLR}^W \cdot \hat{W}_{SLR} \tag{2.10}$$

where ${}^C \hat{J}_{SLR}^W$ is a $6x12$ matrix and $\hat{W}_{SLR}$ is a $12x1$ matrix. After analysing the properties of these matrices, the internal forces can be assumed and calculated by the following formula [2].

$$W_{SLRi} = null({}^C \hat{J}_{SLR}^W) \cdot null({}^C \hat{J}_{SLR}^W)^T * \hat{W}_{SLR} \tag{2.11}$$

where *null* is the nullspace of the matrix. The calculation of equation (2.11) is given in Appendix C. The internal forces can be transform to the left respective right side by projecting the forces to the point L respective R.

$$W_{CL} = {}^C J_{SL}^{-T} \cdot W_{SLi} \tag{2.12}$$

$$W_{CR} = {}^C J_{SR}^{-T} \cdot W_{SRi} \tag{2.13}$$

where W_{SLi} is the first 6 rows of W_{SLRi}. In the same way W_{SRi} is the last 6 rows of W_{SLRi}.

With the information of the internal forces it is now possible to compute the path correction of the left respective right arm in order to minimize the internal forces. This function is in the next chapter implemented to the simulation system and also be prepared for experiments. The implementation of this function can be found in chapter 3.2.1.

2.4 Robot Model

To make it easy to program an industrial robot the use of different coordinate frames is very important. It makes it much easier to get an overview of the instructions and it also allow the environment to be more flexible. Conversion between the coordinate frames is also very important and it turn out to be a simple process. An example of different coordinate frames are given in Figure 5, however not all of them are used in this thesis. Here some frames are defined in relative to another frame, hence the coordinates according to the world frame will therefore be dependent of the exact arm configuration of the robot. Transformations between coordinate frames is an essential part of robot controlling of industrial robots. The most difficult part is the inverse kinematic since this can have infinitive many solutions.

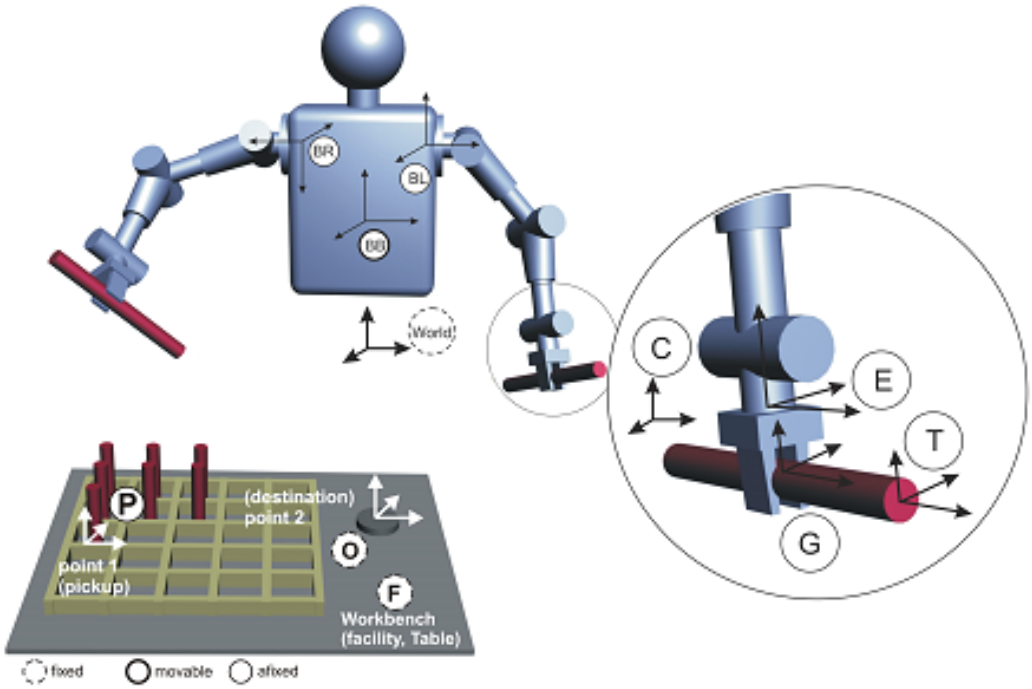

Figure 5: Example of different coordinate frames.

In respect of the motion operation of two arms, the design of the arms is of interest. This in order to be able to do the frame transformations. The left and right arm are identical except that one is mirrored of the other. Each arm has seven DOF in order to be flexible. A model of one of the arm is given in Figure 6 and the units in the figure is in mm.

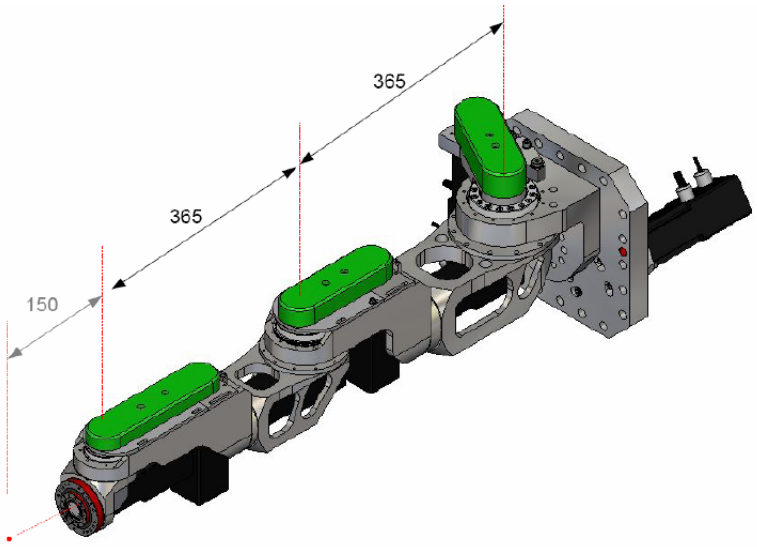

Figure 6: Robot arm configuration.

For the calculation of coordinate frames, the arm has to be split up in different parts where each part is only allowed to rotate around one axes or move along one axes. A such split up of both arms is shown in Figure 7.

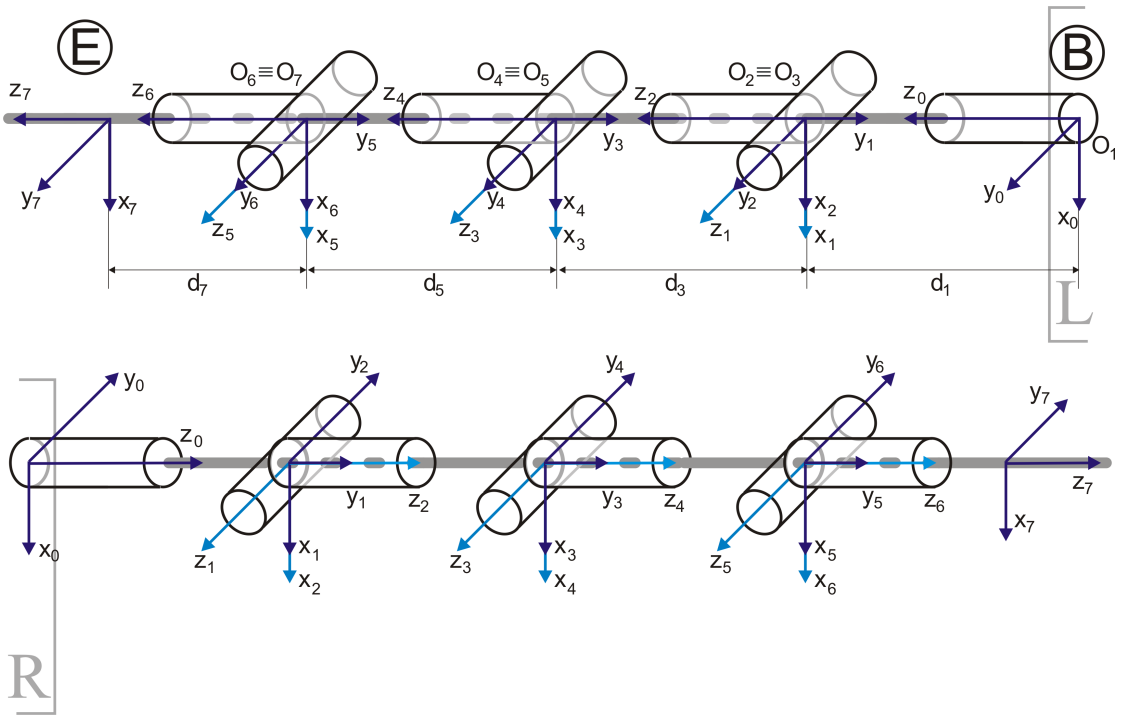

Figure 7: Robot Transformation matrixes.

When calculating the internal forces generated in the beam, the coordinate frames around the gripper is important. The force sensor measuring the forces might not have the same rotation as the beam itself. The different frames are defined in Figure 8 and the conversations between these frames are used in the simulation of the robot.

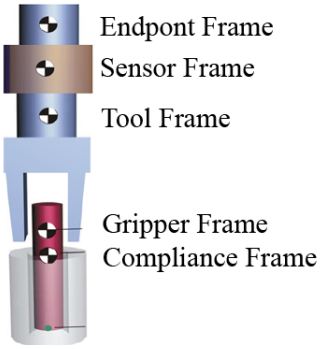

Figure 8: Gripper frames.

A transformation from the position i to the position j can be described with a function ${}^{i}T_{j}$ which is given by equation 2.14 [2].

$$ {}^{i}T_{j} = \begin{bmatrix} {}^{i}R_{j} & {}^{i}p_{j} \\ 0_{1x3} & 1 \end{bmatrix} \tag{2.14} $$

where ${}^{i}R_{j}$ is a rotation matrix and ${}^{i}p_{j}$ is the distance between i and j defined as $[\triangle x, \triangle y, \triangle z]^{T}$. The matrices which are only rotating around one of the axes are defined according to equations 2.15,2.16 and 2.17 [2]. This function has the property that ${}^{i}T_{j}$= ${}^{i}T_{w} \cdot {}^{w}T_{j}$ for any position w.

$$ RotX(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) & 0 \\ 0 & sin(\theta_x) & cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.15} $$

$$ RotY(\theta_y) = \begin{bmatrix} cos(\theta_y) & 0 & sin(\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\theta_y) & 0 & cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.16} $$

$$ RotZ(\theta_z) = \begin{bmatrix} cos(\theta_z) & -sin(\theta_z) & 0 & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.17} $$

The equations and methology described above in this section is enough for transforming the different coordinate frames. The tranfortmation from the Base of the arm (B) to the force sensor frame (S), as seen in Figure 6 is given in equation 2.18. $TransZ(_{\triangle}z)$ is a movement with the length $_{\triangle}z$ along the z-axes.

$$
\begin{aligned}
{}^{S}T_{B} = \; & RotX(\theta_1) \cdot RotY(\theta_2) \cdot TransZ(0.365) \cdot RotZ(\theta_3) \cdot RotY(\theta_4) \cdot RotY(-90^\circ) \\
& \cdot TransZ(0.365) \cdot RotZ(\theta_5) \cdot RotX(\theta_6) \cdot RotX(90^\circ) \cdot TransZ(0.15)
\end{aligned} \tag{2.18}
$$

3

Implementation

The methods used for implementing the theory from chapter 2 has been divided into two sections. First one is about the the programming and implementation for the pi4_workerbot robot and the other one is about the implementation for a simulation model in Simulink. In order to not damage the robot it is essential to test the new controlling algorithms in a simulation environment before the operations are executed on the robot. The simulations provide a technical tool to communicate, visualise and test ideas for the engineers. This will reduce the development cost but also save time as well as help to comply with safety conditions and regulation. It is also an environmental friendly sustainable solution since the computer need much less energy than the robot. Further more it allows several developer to work on the robot at the same time and also during maintenance periods of the robot. Previous work have successfully implemented single arm motion according to safety norm 10218, "requirements and guidelines for the inherent safe design of industrial robots". The extension will be based on this system and also comply with the safety norm.

In order to be able to compare the result, the implemented controller for the robot and the simulation will be the same. The desired control algorithm for controlling the dual-arm robot was decided to be an impedance controller with position mode (Section 2.2.4). The flexibility of the impedance controller was shown to have a big advantageous over the other control methods in the contact case. This because the control of the interaction force is not essential for successfully completing the task. The position based impedance control with an outer force feedback loop was chosen since this model was physical possible to implement on the robot and will be a practical solution for the industry.

Since the system for the robot and the simulation are supposed to be similar, some of the code will be reused for both the simulation and the execution of the robot. From

safety point of view this is also good since it would allow the designer to detect errors and bugs before they are executed on the robot. This can be done by programming the modules in C++. The C++ code could then be integrated to Simulink via a MEX-file loaded by the S-Function in Simulink. MEX is a MATLAB command that compiles and links source files into a shared library, executable from within Matlab and Simulink. The resulting file has a platform-dependent extension (*.mexw*32 for 32 bits Windows, *.mexw*64 for 64 bits Windows and *.dll* in older Matlab versions). For this Eclipse Helios together with MinGW have been used.

3.1 Robot environment

In order to have a fast and robust execution of the robot, a Linux computer is mounted onto the robot that controls the actuators. The programming environment used by the computer is made in C++ and therefore the developed Matlab modules are not possible to directly implement for executing the motion operations on the robot. The different hierarchy level of the programming has been organised according to the figure 9.

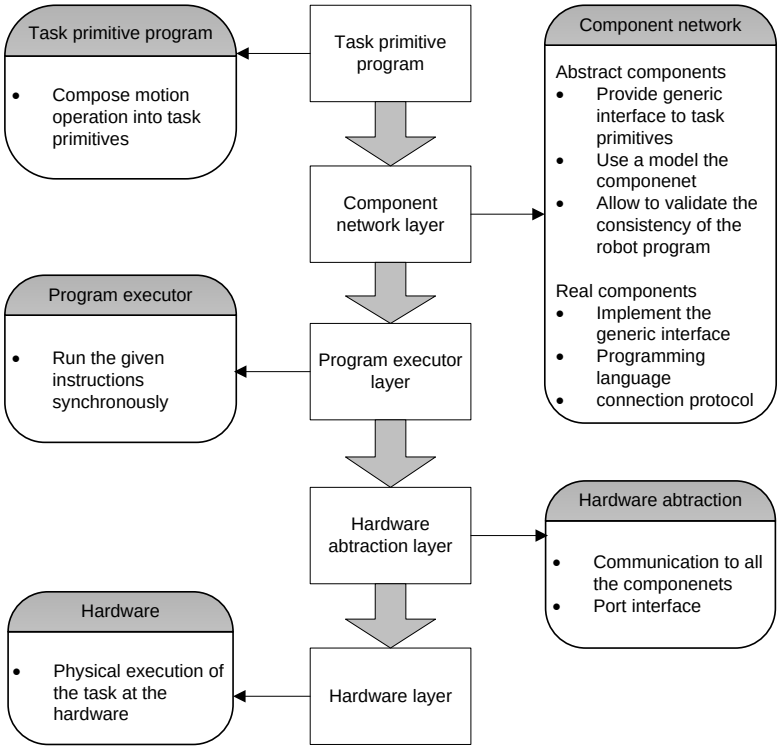

Figure 9: Programming hierarchy level layout.

The top layer, Task primitive, is made for being used by the customers, while the other layers do not need to be visible for them. It is possible for the customer to make many layers above the Task primitive. The module for calculating the internal forces has been programmed to be in the Component Network layer. The designed internal forces function is therefore also written in C++. This function has been developed by the program Eclipse Helios IDE for C/C++ Developers version 3.6.2 platform [17] together with MinGW-w64 [18] and Boost libraries [19].

One of the future possible features for dual-arm robots is an intelligent dual-arm motion programming interface. This would allow an operator to program complex operations via a simple interface. Instead of the operator telling exact how the robots should solve the task, it could be up to the robot to solve the task. For example if the task is to lift a pencil, then only one arm is needed. However if the task is to lift a laptop, then

most certainly, lifting it with both hands is desired. Operations like this opens up for a new programming language for multi arm industrial robotic. For this visual reality could be a very helpful tool since it focus for the operator to describe the desired operation in respect of the object.

It would be a good idea to implement the dual-arm motion algorithms at the Program executor layer which were defined in Figure 9. For this layer, an idea could be to define a semaphore which corresponds to whether a dual-arm operation is taking place or not. Implementing the system in this way could prevent the programmer of the task primitive program of making errors. By activating this semaphore, the system would only allow dual-arm motion operations.

Parameters for the impedance controller could one time in the beginning be initialized, such stiffness, damping, mass as well as compliance frame coordinates. If the robot is working in an unknown environment, it could also be up to the robot to find some of these parameters by moving the object around as well as using image techniques [6].

The implemented function is based on matrix mathematics which is not initially supported in the general C++ language. A Matrix library had previously been developed at IPK which were extended with the functions in order to be able to calculate the necessary matrix operations. The implementation is given in Appendix B. In the chapter 4 the different way of calculating the internal forces is compared.

3.2 Simulation

A simulation environment of the dual-arm robot and the controller have been developed in Simulink. In this model both arms are simulated at the same time and the movement of the arms can be observed by visualisation functions or by looking at the data. Each part of the system in this section is described with focus on the new developed function of the calculations for the internal forces. Single arm simulation had been simulated before this thesis started and also safety conditions had been investigated. The Matlab version used for the simulations have been Matlab 2010b and Matlab 2011a. The model could also be simulated in older versions, but due to a new feature in Simulink some blocks will then be 180 degrees rotated. It is therefore not as easy to get a good overview of the whole system if an old Matlab version is used, however the simulation results will be the same.

Running the simulation with two arms simultaneously is a time consuming task and therefore a new computer with several processors was used. However to fully use the capacity of this computer a few old modules had to be updated from 32 bits to 64 bits. For the Matlab code this is simple and done very quickly, while for the C-functions it is a bit more time consuming task. Due to the lack of time in this thesis all modules except the IPO module were successfully updated and will also work in Matlab 64 bit. As a result of this the full simulation still has to be simulated in with Matlab 32bits, but tests without the IPO will work in the 64bit version.

3.2.1 Overview

The full systems consists of two arm connected parallel. One being set to describe the left arm and the other as the right. The modules which are generating the desired path for the each arm is called IPO (section 3.2.1). The desired path is sent to a SERVO module (section 3.2.1) which task is to compare the actual position of the robot and the desired path from the IPO. From this data the SERVO module will calculate a correction torque Tau. On the robot arm, sensors are measuring the actual positions while in the simulation environment, a model of the robot dynamic has to be used. This dynamic module (section 3.2.1) has the torque as input and return as an output the angles and angular velocity of the joints for each arm. It is important that the IPO and the Dynamic modules is set with the same initial values in the beginning in order for the simulation to be reliable.

The position of the two arms is being observed and the relative distance between the arms will be used by the external object module. For the simulation an external object as a beam has been developed (section 3.2.1). When the arms have gripped the beam forces will be generated depending whether if it is compressing or decompressing. The calculated forces (or measured forces in the real system) is then used by the internal force module (section 3.2.1) using the formulas of section 2.3. The internal forces will

afterwards be feed to the IMCO module that calculates a path correction matrix that are being feedback to the IPO module. Both arms servo modules can use the same Supervisor module since this signal is going to be identical during the simulation. The overview of the system is shown in Figure 10.

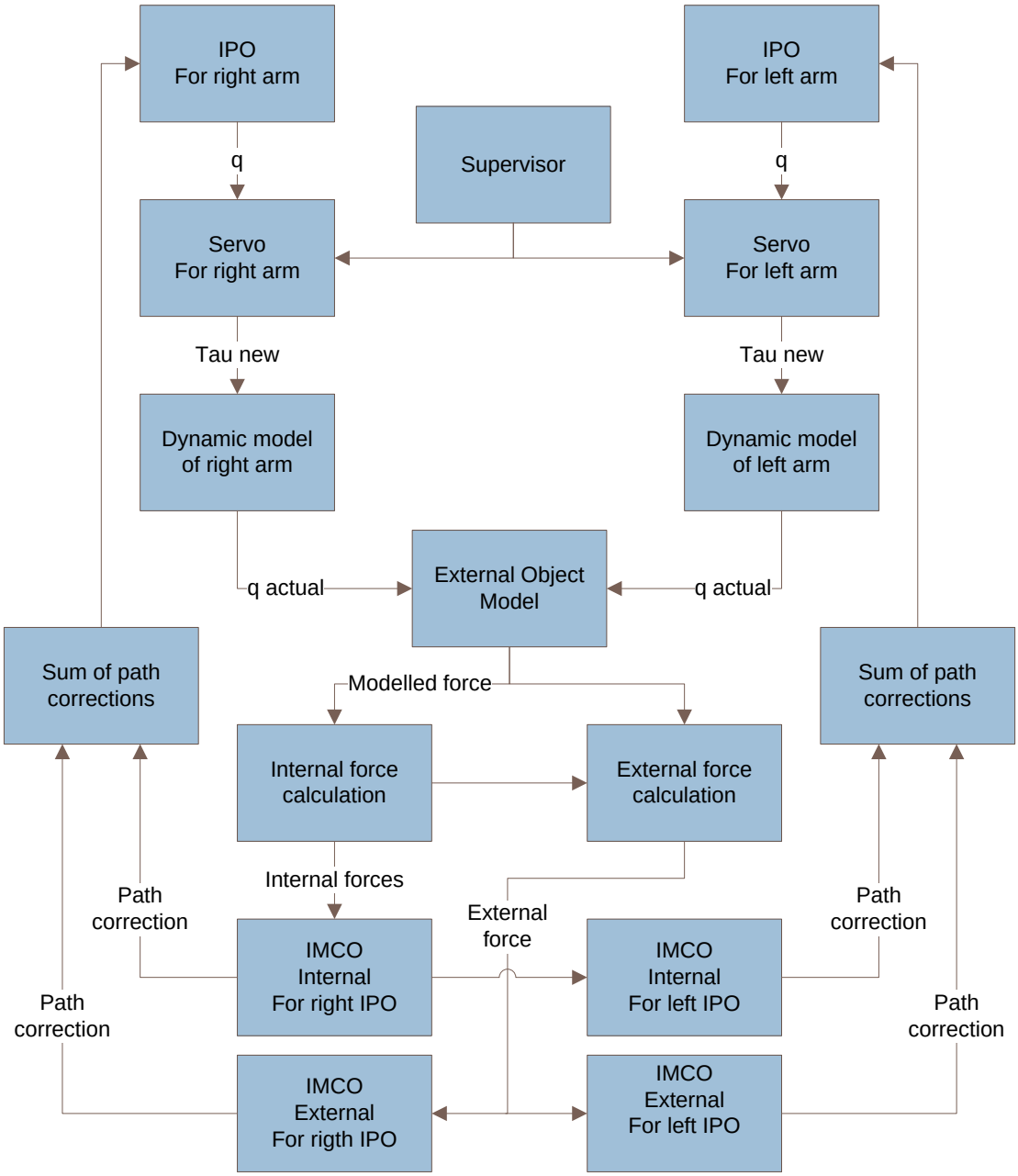

Figure 10: Overview graph of the dual-arm simulation model.

IPO module

The IPO Module task is to generate a path for the arm which it is assigned to, either the left or the right arm. IPOs desired path can be set in several different ways; PTP (Point to Point), CP (Continuous Path) and JOG (which is used lock the position of the joint axes or to limit the movements to some dimensions). The IPO have several different states which is used during the initialization and then the state running is active during the movement operation. The state Auto will go over the necessary states in the beginning and go over to monitoring when the arm have completed all the tasks and reached its goal. One important setting is the path correction which is described in section 3.2.1.

Most of the IPO module is programmed in C++ which mean that the same code can be used for the execution of the real robot. The module is quite complex and its libraries have to be compiled before the MEX is being compiled. Since this thesis is focusing on the internal force calculations of the simulation, the MEX file will therefore be treated as a black box. The S-block S_Robotarm has two parameters which have to be set. One is the whether it is desired to module an IPO for left or right arm and the other is the initial values of the joints.

SERVO module

The Servo module task is to compare the actual position of the robot and the desired path from the IPO Module. As an output the Servo gives a torque that will be sent to the actuators so that the arms will move. Inside the Servo, PID controllers are controlling the joints to follow the desired paths. The servo is being initialized by a supervisor whose task is to go over the different state in a similar way as the Auto option for the IPO. The Servo is as the IPO module made in C++ in integrated via a MEX-file. It will also be considered as a black box in this report. For an overview of the Servo see Figure 11.

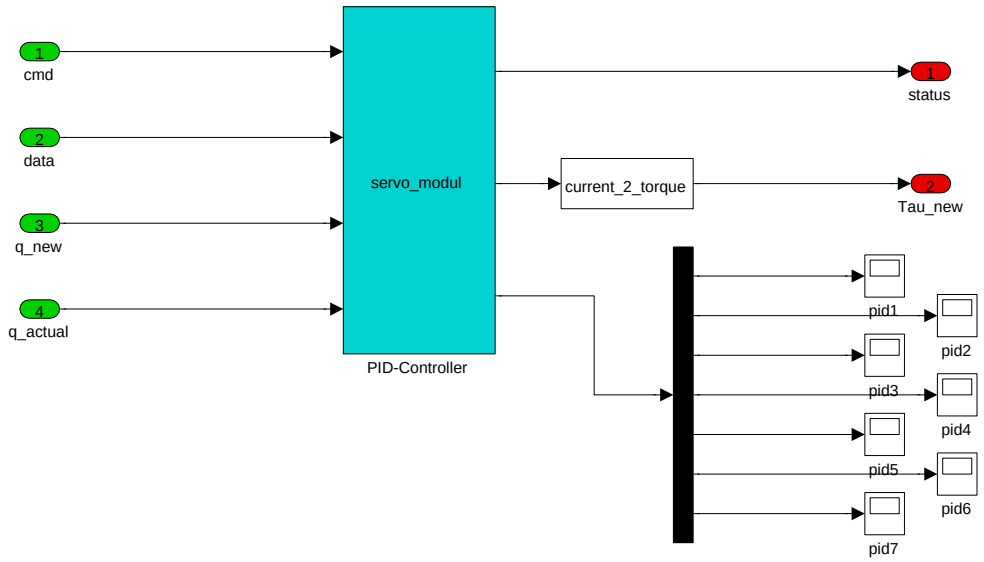

Figure 11: Overview of the Servo Module.

DYNAMIC module

The dynamic module is a model of the robot dynamics which is only used for the simulations. There are several non linear constraints which makes it difficult to control everything in the IPO. For example the actuators can only handle a certain speed, friction can prevent the arms from moving a slow speed and also singularies can cause problems when one of the robot arms is a bit away from its path. The dynamic module includes gravity, friction, coriolis, centrifugal as well as the case when some breaks have been locked into a certain angle. As of the experiments of the with the internal forces, it is not necessary to have activated the dynamic model at all if the holding of beam is instead made by the data from the IPO. When the internal forces is evaluated, the dynamic model could also be tested to work during contact operation.

External object module

An external object modeled as a beam has been analyzed. In this beam module the stiffness parameter K is varied while the damping coefficient d is assumed to be zero. The input variables of the coordinates can come from two different sources in the simulation while in the execution of the robot it will always be read by the force sensors. In the simulation the data can be either be read from the Dynamic module or directly from

the IPO. The accuracy the dynamic model is not yet verified for the case of of contact operations and therefore the data from the IPO will instead be used. The location of the compliance frame C is assumed to lay in the middle between the two arms. The gripping of the beam has been decided to be so that the z-axis of the S-frame lays in the direction of the beam.

Internal force module

The module from section 2.3 can be calculated using a MATLAB function (Appendix A) or using a MEX-File (Appendix B) which is the code when executing the motions for the robot. The two ways of calculating can be switched inside the simulation module, and they both gives similar results as expected. The rotation matrices which are given as input is also calculated before the modules are called. The input data to the functions are summarized below.

- The coordinates of L, R and C.
- The rotation matrices between L and C (Rot_LC) respective between R and C, (Rot_RC).
- The forces measured at L and R, (W_LR)

IMCO module

The IMCO (Impedance Controller) module task is to generate a path correction. The input of this module is the applied force in respect to the sensor frame. The IMCO is as well made in C++ and will be considered as another black box. As an input, 6 different controllers can to be decided, for the coordinates as well as the rotation part. The parameters could be set to low/medium/high damping or low/medium/high stiffness.

A correction matrix is from this data as well as the TCP calculated and given as an output. Only the first 3 rows of the 4 rows is being communicated since the last row is a constant with the values $[0, 0, 0, 1]$. A special note should also be made for the correction matrices. When the module has several correction matrices it is not strait forward to add the path corrections together with each other. The position should be superpositioned and for the rotation part it is a bit more difficult. In the case of sum, the new rotation is given as the product of the two rotation matrices.

3.2.2 Visualisation

In order to validate the performance of the robot and understand the movement of the arms of the simulation, several different ways of visualize the robot looking at the movement have been developed. One of these is an animation module made in Simulink virtual reality box. In this thesis this animation module has not been included, but the developed simulation environment is compatible with also this module. This mean that

the simulation can save the paths of the arms, which afterward could be used as an input to the animation module which generate a video of the motion. For verifications of the developed functions, instead simple graphs of the movements of the joints and the endpoint location of the arms have been used.

The graphs used for visualization can also be used together with data from the experiments. In such way the simulation environment is being verified and the data can be better analyzed. In the next chapter the results from the simulations can be observed.

4

Results

The experiment to be analyzed was in beginning of the report decided to be Bi-Hold which is defined in Figure 2. This because this operation is basic and that many other bimanual operations are dependent on this operation. Before anything can be executed for the real robot it is very important that this motion operation is safe and that the simulations are robust. The parameters of the control algorithm have to be tuned for best performance without jeopardize stability. For the simulation of Bi-Hold an elastic external object was chosen and the speed was initially set to be relative slow. Different stiffness of the controller as well of the object will be simulated.

The movement is therefore first thoroughly analyzed by the simulations. In order to save simulation time the movement was decided to be very small and that the beam would be compressed. The arms are starting slightly outside the sides of the beam to a point where the distances between the arms are shorter than the actually beam. When the arms then are in position over the beam they grip the beam so that forces are applied from both sides to the beam. The motion will continue and compress the beam so that the internal force algorithm will calculate and generated internal forces within the beam. This data is afterwards send this to an IMCO module which task is to generate a path correction for the arms. The path correction is given to the IPO module which will correct the path. The reaction of this closed loop system is seen to react fast enough to compensate for the internal forces. Below are figures showing the simulations of the analyzed motion. To show how small the desired motion, a simulation without gripping the beam is shown in Figure 12.

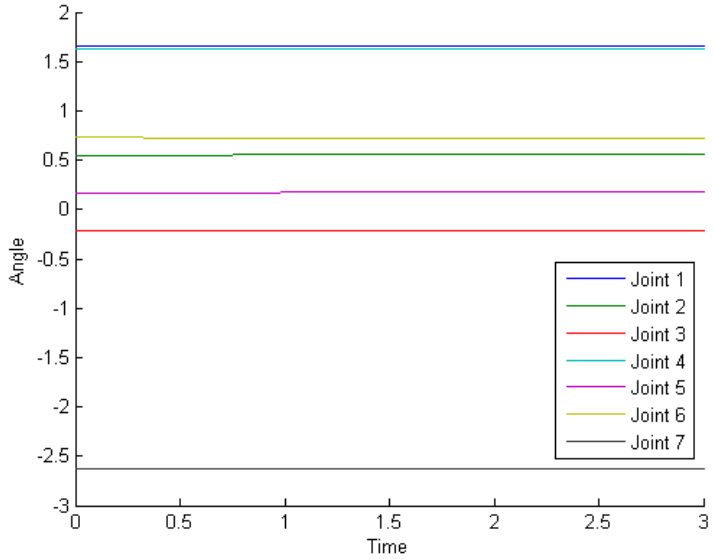

Figure 12: Simulation of a small movement without gripping the beam.

As can be seen of Figure 12 the change of values of the joints are very small, almost constant during the whole motion.

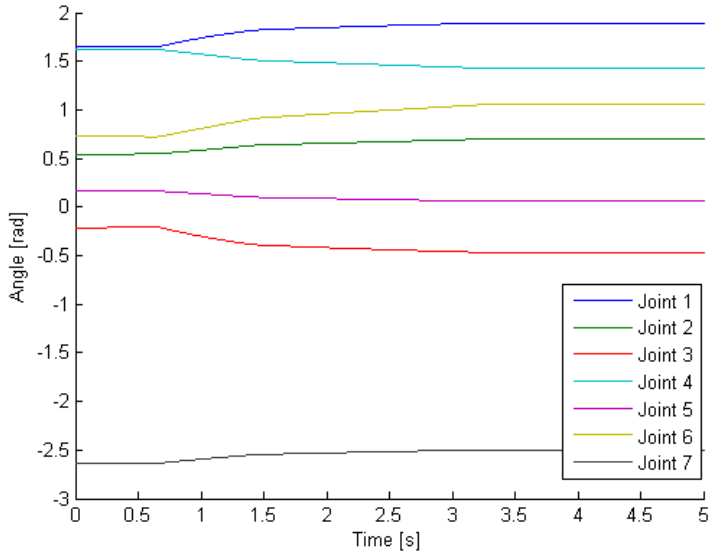

Figure 13: Simulation of a small movement with gripping the beam.

As can be seen the changes of the values of the joints in Figure 13 changes a little bit compared with the case of not gripping the beam. However, when looking at the endpoint location of the arm, the position is almost the same and does not move much

at all. The system manages to to compensate for the internal force and correct the path. How fast the system manage to control the internal forces can be set with a parameter K which corresponds to the stiffness of modeled beam. In figure 14 the simulation has been executed with different values of K. The force in this diagrams corresponds to which force the controller thinks the system has, the compression of the beam is inverted proportional to the modeled force with the factor $1/K$. In the simulations an 8cm long beam has been simulated. As can be seen from the graphs the system reduces the internal forces in around a half second. The chosen impedance controller has been set with low damping and it should also be noted that the IMCO is not aware of the stiffness of the beam.

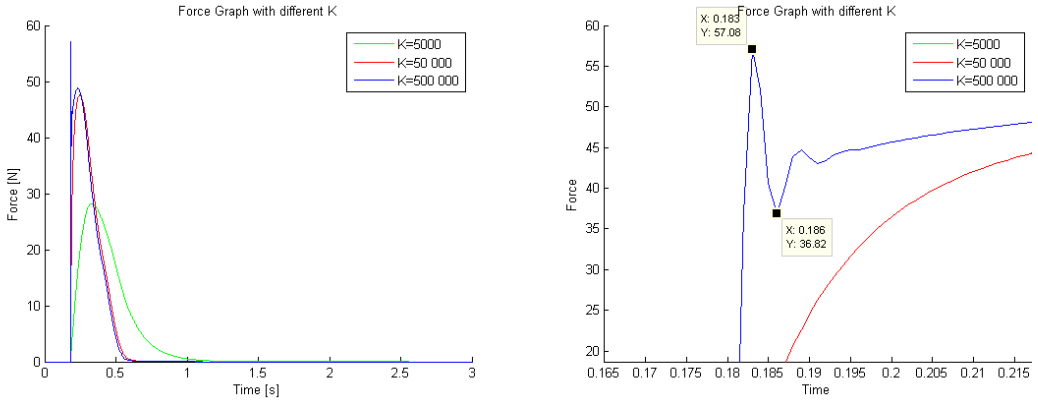

Figure 14: Internal force data from simulations with different values of K.

On the right side of Figure 14 the peak has been zoomed. Here it is being seen that the contact generate the peak when the system is simulated with $K = 500\ 000$. Since the force afterwards does not go down more than to 36N after the peak, it is not a problem. However if K is would be set very stiff, then a drop of the force could make the force go down below zero. In the case where the object is only being held and not gripped between the arms this could lead to that the object is dropped. If the arm would hit a surface it would mean that the arms would bounce a few times before a stable contact been archived. The path correction calculated by the IMCO is given in figure 15.

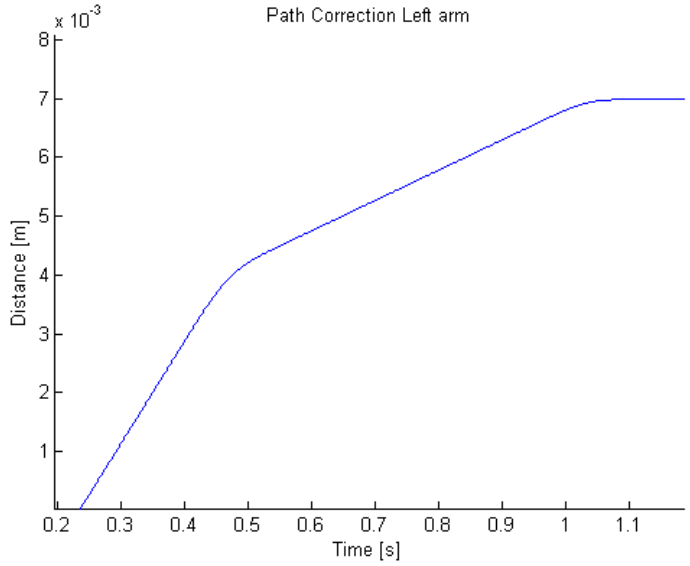

Figure 15: Path correction for the left arm with $K = 500\ 000$.

The internal force calculation routine in C++ was also evaluated. In Figure 16 the error between the calculations of internal forces in Matlab respective C++ is given. As can be seen the difference is less than $10^{-14}N$. The most accurate function is most likely Matlab, however the difference is much smaller than the noise that would be seen in practice.

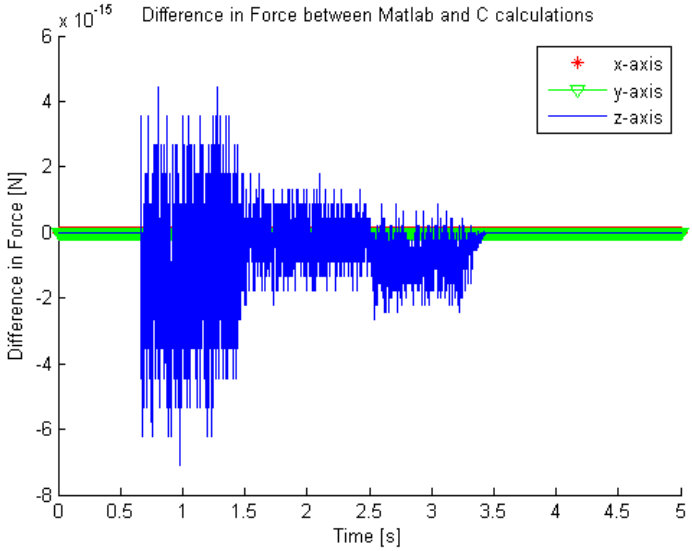

Figure 16: Comparsion between internal forces calculated by C++ and Matlab..

Different speeds of the arms were tested, to see how the system would react when one arm is faster than the other. In this simulation which is given in Figure 17 the right

arm finishing its desired goal before the left arm.

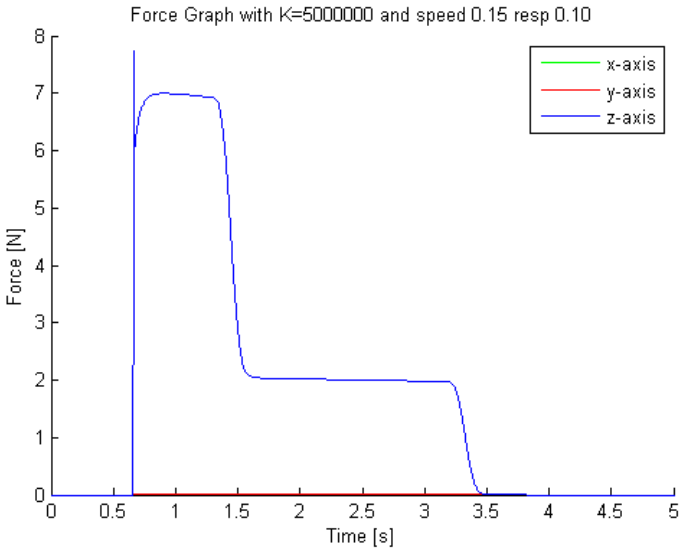

Figure 17: Simulation with different speeds between the arms.

Finally also a free space motion was tested together with the algorithm. The motion is now changed from stating called safe position of the robot. This is when the arms are stretched out far away from each other. The desired goal is the same as before and also the length of the beam. As can be seen there is no strange behavior due to the contact control algorithm. In the case where an impedance control with force mode is used the position could had been affected [2].

5

Discussion

The development of dual-arm robots is currently a hot topic within the robotic industry. Many dual-arm robot concept have been presented by different companies even though the dual-arm concept is not yet widely implemented within the industry. A bi-manual operation for a dual-arm robot holding a beam have been investigated. For this a simulation environment of the robot was built up which is a good approach for developing functions to new systems. It should be noted that the simulation environment needs more information than the real robot and that the all functions have to be executed on the real robot before they are verified. A controller called impedance controller with position mode was selected for the simulations of holding a beam and this control method turned out to give very satisfactory results for this operation in the simulations. The execution of this controller has not yet been done at the robot and therefore the function is not yet verified.

The controller worked good in the transition between free space and contact with an external object. Uncertainties in the positions were also no problem and internal forces in a beam could also be compensated within 0.5 seconds after contact. The controller also does not make any undershoot of the force which mean that the contact is stable between the manipulator and the external environment. This mean that the arms are not bouncing on the surface after contact. Simulations were only simulated when the beam tried to hold the object still and not when the object is moving. The selected control method is not necessary the best control algorithm for all the assembly options. Each operations have to be investigated and it should also be noted that there is no problem to use different control algorithm for different assembly operations.

The gripping of the object could be done in several different way. It is very important that this is being observed since it will change the behavior of the control system and mistakes could damage both the external object as well as the robot. Where and

how to grip external the object is also a parameter which could be optimized. Since the arms have 7 DOF each, there also exist a subspace of position solutions which the arm could have when gripping the object. The best way to grip the object is dependent on the operation that should be solved as well as upcoming instructions. It should be emphasized that the robot can solve many of the instructions in many different ways and that it is not critical if a non optimal path sometimes is used.

6

Conclusion and Recommendations

The benefits of a dual-arm robot compared with a human operator or single arm robot were multitasking and space saving. The tasks that the dual-arm will perform will most likely be very similar to motions that the human is capable of. The fundamental bimanual motions for assembly was analysed to understand the tasks of the dual-arm robots.

The purpose of this thesis was to research dual-arm motion planning and to develop a control algorithm for a dual-arm robot during a contact interactions. One of the basic dual-arm motion, Bi-Hold was successfully implemented and tested in the simulation environment with satisfactory result. The impedance controller managed to control the arms in such a way that the internal forces of the beam was decreasing and approaching zero after about 0.5 seconds after impact without making any force undershoot.

A module for implementing the same function for the robot was developed and was also simulated to work satisfactory but it has not been verified by real experiments on the robot. The algorithm works well when the arms are moving in the free space as well as during the contact phase. However, only one of the bimanual motions have been investigated. This thesis have created a wide base for further research and the author of this thesis recommend the following further investigation.

- Practically verify the internal force calculations also on the Pi4_workerbot.
- Implementing the other Bi-manual motion operations that have been defined in Figure 2.
- Develop a safe dual-arm motion operations interface within the program component layer.
- Implement a model of the grippers into the simulation environment and prove ensure that an external object is not dropped.

Bibliography

[1] Pi4_robotics, Arbeitsroboter: "pi4_workerbot" (May 2011).
URL http://www.pi4.de/ds571.html

[2] Y. E. M. Vukobratovic, D. Surdilovic, D. Katic, Dynamics and Robust Control of Robot-environment Interaction, 1st Edition, World Scientific, 2009.

[3] Yaskawa, Motoman sda10d assembly robot (May 2011).
URL http://www.motoman.com/products/robots/models/sda10d.php

[4] KUKA/DLR, Humanoid-robot-justin-learning-to-fix-satellites (Jun. 2011).
URL http://spectrum.ieee.org/automaton/robotics/industrial-robots/humanoid-robot-justin-learning-to-fix-satellites

[5] ABB, Frida concept robot (May 2011).
URL http://www.abb.com/cawp/abbzh254/8657f5e05ede6ac5c1257861002c8ed2.aspx

[6] R. Brooks, Remaking manufacturing with robotics (2011).
URL http://fora.tv/2009/05/30/Rodney_Brooks_Remaking_Manufacturing_With_Robotics

[7] S. C. de Oliveira, The neuronal basis of bimanual coordination: recent neurophysiological evidence and functional models, Acta Psychol (Amst) 110 (2002) 139.

[8] S. G. Tzafestas, Automation, humans, nature, and development, in: S. G. Tzafestas (Ed.), Human and Nature Minding Automation, Vol. 41 of Intelligent Systems, Control and Automation: Science and Engineering, Springer Netherlands, 2010, pp. 1–21.

[9] H. E. Price, The allocation of functions in systems, Human Factors 27 (1) (1985) 33–45.

[10] J. Lee, N. Moray, Trust, control stategies and allocation of function in human-machine systems, Ergonomics 35 (10) (1992) 1243–1270.

[11] R. Bonitz, T. Hsia, Internal force-based impedance control for cooperating manipulators, Robotics and Automation, IEEE Transactions on Robotics and Automation 12 (1) (1996) 78 –89.

[12] Fraunhofer, Institut für produktionsanlagen und konstruktionstechnik (May 2011).

[13] S. C. d. O. O. Donchin, E. Vaadia, Who tells one hand what the other is doing: the neurophysiology of bimanual movements, Neuron 23 (1999) 15–18.

[14] T. Glad, L. Ljung, Control Theory: Multivariable and Nonlinear Methods, CRC, 2000.

[15] S. Schneider, J. Cannon, R.H., Object impedance control for cooperative manipulation: theory and experimental results, Robotics and Automation, IEEE Transactions on Robotics and Automation 8 (3) (1992) 383 –394.

[16] J. De Schutter, H. Bruyninckx, W.-H. Zhu, M. Spong, Force control: A bird's eye view, in: B. Siciliano, K. Valavanis (Eds.), Control Problems in Robotics and Automation, Vol. 230 of Lecture Notes in Control and Information Sciences, Springer Berlin / Heidelberg, 1998, pp. 1–17.

[17] Apache-Software-foundation, Eclipse helios (May 2011).
URL `http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/heliossr2`

[18] Sourceforge.net, Mingw-w64 (May 2011).
URL `http://sourceforge.net/projects/mingw-w64/`

[19] Sourceforge.net, Boost (May 2011).
URL `http://sourceforge.net/projects/boost/files/boost/1.46.1/`

A

Source code of internalforce in MATLAB

```
function [W_CI ] = internalforces(L, Rot_LC, R, Rot_RC, C, W_LR)
% Calculations module of the internal forces
% Author: Daniel Andersson
% Last Modification: 7th May 2011

%% Calculations of vectors

Δ = C - L;
x = Δ(1); y = Δ(2); z = Δ (3);
LCx = [0 -z y; z 0 -x; -y x 0];

Δ = C - R;
x = Δ(1); y = Δ(2); z = Δ (3);
RCx = [0 -z y; z 0 -x; -y x 0];

%% Calculations of jacobians

Jac_LC  = [ Rot_LC, -Rot_LC * LCx ; zeros(3,3), Rot_LC];
Jac_RC  = [ Rot_RC, -Rot_RC * RCx ; zeros(3,3), Rot_RC];
Jac_LRC = [ inv(Jac_LC )  inv(Jac_RC )];

% Generalized Jacobian
Gen_Jac_LRC = Jac_LRC *inv(Jac_LRC*Jac_LRC );

%% Calculations of Wrench at T
W_T = Jac_LRC * W_LR;

%% We need to calculate the nullspace
Nullspace_Jac_LRC = null( Jac_LRC); %
```

```
30 lambda = Nullspace_Jac_LRC * W_LR;
31
32 %% Time to internal forces
33 W_LRexternal = Gen_Jac_LRC*W_T;
34 W_LRinternal = Nullspace_Jac_LRC*lambda;
35
36 W_LRtotal = W_LRexternal + W_LRinternal;
37
38 %% Calculation of Wrenches
39
40 W_CL = inv(Jac_LC )*W_LRinternal(1:6);
41 W_CR = inv(Jac_RC )*W_LRinternal(7:12);
42
43 W_CI = [W_CL;W_CR];
44
45 end
```

B

Source code of internalforce in C++

```
1  /* ********************************** */
2  /* Author : Daniel Andersson          */
3  /* Date   : 17.04.2011                */
4  /*                                    */
5  /*  Description:                      */
6  /*  This is does the same as the      */
7  /*  internalforces.m but instead in C.*/
8  /*                                    */
9  /* ********************************** */
10
11
12 // Project includes
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include <string.h>
16 #include <math.h>
17 #include "internal_force_calculation.h"
18
19 // MATLAB Includes
20 #ifdef  MATLAB_MEX_FILE
21     #define M_PI 3.14159265356
22     #include "mex.h"
23     #include <simstruc.h>
24 #endif
25
26 //Sufficient libmaths includes
27 #include <stdio.h>
28 #include <stdlib.h>
29 #include <iostream>
```

```
30 #include <stdio.h>
31 #include <stdarg.h>
32 #include "matrix.h"
33 #include "vector.h"
34 #include "svd.h"
35 #include "qr.h"
36 #include "algebraicoperations.h"
37 #include "mathexceptions.h"
38
39 using namespace LibMath;
40
41 INTERNALFORCE::INTERNALFORCE()
42 { // Constructor
43
44 // Initialize:
45   for (int i = 0; i < 12; i++) {
46     W_SLRi[i] = 0;
47   }
48
49 }
50
51 void INTERNALFORCE::calculate(double* L, double* Rot_LC_, double* R,
52                           double* Rot_RC_, double* C, double* W_SLR_)
53 {
54
55     // Variable transformation
56     Matrix W_SLR = Matrix(12, 1);
57     double *data_WSLR = W_SLR.getData();
58     for (int i = 0; i < 12; i++) {
59         data_WSLR[i] = W_SLR_[i];
60     }
61     Matrix Rot_LC = Matrix(3, 3);
62     Matrix Rot_RC = Matrix(3, 3);
63     double *data_Rot_LC = Rot_LC.getData();
64     double *data_Rot_RC = Rot_RC.getData();
65     for (int i = 0; i < 9; i++) {
66         data_Rot_LC[i] = Rot_LC_[i];
67         data_Rot_RC[i] = Rot_RC_[i];
68     }
69
70     // Help variable declaration
71     double x = 0;
72     double y = 0;
73     double z = 0;
74     Matrix Zeros3x3 = Zeros(3,3);
75
76     // Calculations of vectors
77     x = C[0]-L[0];
78     y = C[1]-L[1];
79     z = C[2]-L[2];
80
81     double data_test3x3LCX[] = {0,-z,y, z, 0,-x, -y, x,0};
82     Matrix LCx = Matrix(3, 3, data_test3x3LCX);
```

```
83
84      x = C[0]-R[0];
85      y = C[1]-R[1];
86      z = C[2]-R[2];
87      double data_test3x3RCx[] = {0,-z,y,z,0,-x, -y, x,0};
88      Matrix RCx = Matrix(3, 3, data_test3x3RCx);
89
90      // Calculations of jacobians
91      //  [ Rot_LC, -Rot_LC*LCx ; zeros(3,3), Rot_LC];
92      Matrix a1 = matMatMult(Rot_LC,LCx);
93      Matrix a2 = scalarMatMult(-1,a1);
94      Matrix Jac_LC  = extend_Matrix(Rot_LC, a2,Zeros3x3,Rot_LC);
95
96      //[ Rot_RC, -Rot_RC  RCx ; zeros(3,3), Rot_RC];
97      Matrix b1 = matMatMult(Rot_RC,RCx);
98      Matrix b2 = scalarMatMult(-1,b1);
99      Matrix Jac_RC  = extend_Matrix(Rot_RC, b2,Zeros3x3,Rot_RC);
100
101     //[ inv() inv(transpose(Jac_RC))];
102     Matrix g1 = matMatMult(Rot_RC,RCx);
103     Matrix g2 = scalarMatMult(-1,g1);
104     Matrix Jac_RC_Tinv  = extend_Matrix(Rot_RC, Zeros3x3, g2, Rot_RC);
105
106     //[ inv() inv(transpose(Jac_LC))];
107     Matrix h1 = matMatMult(Rot_LC,LCx);
108     Matrix h2 = scalarMatMult(-1,h1);
109     Matrix Jac_LC_Tinv  = extend_Matrix(Rot_RC, Zeros3x3, h2, Rot_RC);
110
111     // Jac_LRC = [ inv(Jac_LC ) inv(Jac_RC )];
112     Matrix Jac_LRC  = extend_Matrix_vertical(Jac_LC_Tinv,Jac_RC_Tinv);
113     Matrix Jac_LRC_H= extend_Matrix_horizontal(Jac_LC_Tinv,Jac_RC_Tinv);
114
115     // We need to calculate the nullspace
116     //Nullspace_Jac_LRC = null( Jac_LRC);
117     Matrix Nullspace_Jac_LRC = NullSpace(Jac_LRC_H);
118
119     // First step is to calculate lambda.
120     Matrix lambda1 = transpose(Nullspace_Jac_LRC);
121
122     Matrix lambda = matMatMult(lambda1, W_SLR);
123
124     // Now we are readdy to calculate the internal forces.
125     Matrix W_SLR_i = matMatMult(Nullspace_Jac_LRC , lambda );
126
127     // Lets project the internal forces to L resp C.
128     Matrix W_SL_i = cut_out_Matrix(W_SLR_i, 0, 5, 0, 0);
129     Matrix W_SR_i = cut_out_Matrix(W_SLR_i, 6, 11, 0, 0);
130     Matrix W_CL_i = matMatMult(Jac_LC_Tinv,W_SL_i);
131     Matrix W_CR_i = matMatMult(Jac_RC_Tinv,W_SR_i);
132
133     // We are done, lets now write the result to a normal variable
134     Matrix W_CL_iW_CR_i  = extend_Matrix_vertical(W_CL_i,W_CR_i);
135
```

```
136     double* data = W_CL_iW_CR_i.getData();
137     for (int i = 0; i < 12; i++) {
138         W_SLRi[i] = data[i];
139     }
140 
141     return;
142 }
143 
144 
145 double* INTERNALFORCE::get_internal_force(void)
146 {
147     return this->W_SLRi;
148 }
```

C

Mathematical derivation of the internal forces formula

With the matrices given in (2.10), the following equation for the forces can be written to C.1. Here the idea is that the first and second parts are orthogonal.

$$\hat{W}_{SLR} =^{C} J_{SLR}^{W\#} \cdot W_T + null(J_{SLR}) \cdot \lambda. \tag{C.1}$$

where $null$ is the nullspace, $J^{\#}$ is the pseudoinverse

$$J^{\#} = J^T (J \cdot J^T)^{-1} \tag{C.2}$$

and

$$\lambda = null(J_{SLR})^T * \hat{W}_{SLR} \tag{C.3}$$

From this the internal forces is calculated.

$$W_{SLRi} = null(J_{SLR}) \cdot \lambda \tag{C.4}$$

With (C.3) in (C.4), the follow formula is obtained;

$$W_{SLRi} = null(J_{SLR}) \cdot null(J_{SLR})^T * \hat{W}_{SLR} \tag{C.5}$$