Advanced Teleoperation with Haptic Feedback

Eric Ragnarsson

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6153
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 2022

Abstract

As the world moves more and more towards automation of repetitive and dangerous tasks there are still many tasks that primarily involve interaction with an object that are still carried out by humans. This is in many ways due to difficulties in programming the robot to achieve similar performance. The most natural way of transferring these skills to the robot is by demonstration. This can be achieved by the use of haptic feedback to allow the operator to feel the robotic system during the demonstration.

This master thesis seeks to implement and improve master-slave coordination using virtual constraints for a redundant dual-arm haptic interface. This is done by implementing a general-purpose algorithm compatible with different types of 6 degrees of freedom robots in Python. For this project a UR5e robot was used as well as a second virtual clone of the UR5e robot to serve as master and slave arm respectively.

The virtual second UR5e robot was simulated using both Maplesim and inverse dynamics. The inclusion of force-torque (F/T) sensors was implemented and tested.

Free-space motion where position and orientation offsets are retained as well as collision with a soft surface resulting in a feeling of force upon interaction with a soft object were achieved. The singularity-free operation produced consistent and intuitive results, with the inclusion of a teaching handle to improve adjustments. However, low sampling frequency of the implemented algorithm resulted in delays that negatively impacted the translational and rotational error between the end effectors of these robots, as well as not reaching a realistic and intuitive feeling of force upon interaction with a soft object. These problems were not solved due to lack of time.

The impact of the implemented F/T sensor was investigated, but due to the limited time left was this not enough to reach a definitive conclusion.

The implemented algorithm functions well during slower movements, with room for improvements in the optimization of calculation times and further investigation of the impact of the implemented F/T sensor.

Acknowledgements

This master thesis would not be possible without Dr. Mahdi Ghazaei and Professor Anders Robertsson whose help and knowledge helped me immensely in this endeavour. I also wish to thank the wonderful people at Cognibotics who helped me with my many questions about the company's system and code. To Cognibotics, I would like to extend my warmest thanks for the opportunity to learn from my time with you.

Contents

1

Introduction

The use of robots has increased and expanded drastically from the industrial robots since early 1970 [“History of Industrial Robots” 2012]. With the increased use of autonomous robots, there is still an increased need for interactive feedback. For instance, one of the remaining difficulties in programming has been to program robots to perform certain tasks that involve force interaction. One of the solutions to this is to have the operator demonstrate the task.

Another thing that can help further is the introduction of a haptic interface. This will ease the programming difficulties of tasks that involve force interaction by providing the operator with a way to act and sense through the robot [Sang Hyoung et al., 2015].

Another example of areas where a haptic interface will improve the interaction with robots is in the field of teleoperation. Here an haptic interface opens up easier and more intuitive ways to perform teleoperated tasks such as micromanipulation and microassembly. These teleoperation systems are today widely used within these fields where an assembly is controlled through a joystick [Bargiel et al., 2010], but is something that makes the process less intuitive.

There are still hurdles that have to be addressed. These include uncertainty of friction forces and other parameters, communication delays, nonlinearities which result in challenges for stability and robustness of the haptic system. Furthermore is the improvement of safety a concern, as the robots share their workspace with humans.

This thesis seeks to implement an algorithm resulting in a safe and functional haptic interface between two robot arms in Python.

1.1 Goals of thesis

The overarching goal of this project is the implementation and improvement of a dual-arm haptic system in Python. The haptic system used for this thesis will be implemented on two UR5e robot arms.

The UR5e is a medium sized robot designed by Universal Robots. It has 6 joints, a payload capacity of 5 kg and a reach of 850 mm [*UR5e Information* 2020]. The robot arm used can be seen in Figure 1.1.

The second robot arm was implemented as a virtual robot, visualisation of this robot next to a visualisation of the real robot can be seen in Figure 1.2. The improvements include the inclusion of an F/T sensor and a handle for ease of adjustment. The two main objectives are

- Creation of a reliable haptic interface
- Enhancing the user experience

The expected outcome of this thesis project is to create a haptic interface with a designed teaching handle. This haptic interface will perform to the following specifications

- **Performance**
  - Singularity-free operation.
  - Free-space motion where position and orientation offsets are retained.
  - Consistent and intuitive results of input signals.
- **User experience**
  - Ability by the user to use the interface to set position and orientation offsets without extensive programming.
  - A realistic and intuitive feeling of force upon interaction with an object.
  - Workable teaching handle to improve adjustments of the remotely controlled (slave) arm.

Demarcation

Due to the limited time of this thesis the following demarcation is necessary.

- The impact of the implemented F/T sensor was not investigated in great detail.
- The ability of the implemented algorithm to react to the external torque applied to the joints of the robots was not implemented.

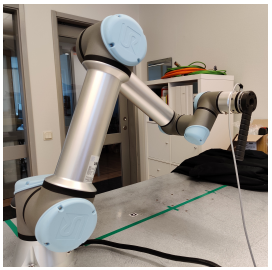


Figure 1.1 The UR5e robot arm that was used for the implementation of the dual-arm haptic system. The F/T sensor and handle is included. This robot arm was the baseline for the second virtual robot marked in red head in Figure 1.2.

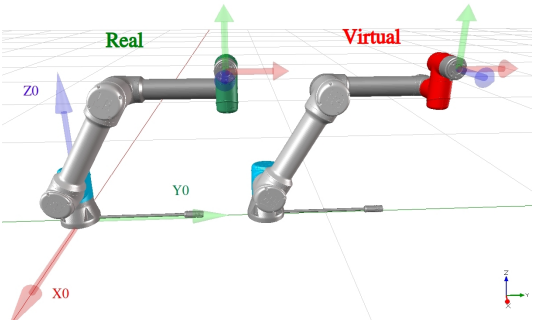


Figure 1.2 Visualisation of both robots used for the implementation of the dual-arm haptic system. The UR5e robot with a green head is a visualisation of the real robot and the robot with a red head is a visualisation of the virtual robot. Both have been included for comparison. The virtual robot is placed 0.75 m from the real robot in the Y0-direction.

1.2 Outline of thesis

Chapter 2 contains the background for the project and its connection to previous work. In Chapter 3 are the theories and mathematical formulas that are used and implemented into the algorithm presented. Chapter 4 explains the modelling and simulation of a robot arm for use as a second virtual robot. The experimental setup is covered in Chapter 5. The tests used to evaluate the implementation of the algorithm are described in Chapter 6. The results of the evaluation of the implementation and the discussion of these results can be found in Chapter 7. Chapter 8 discusses the conclusions of the project and its continued work.

The dimensions for use in the creation of the simulated robot arm, more detailed information on the construction of the simulated robot, the sensitivity report for the F/T sensor as well as the coordinate system used for calculating the homogeneous transformation matrices can all be found in the Appendix.

1.3 Company which offed the thesis work

The thesis work has been conducted at Cognibotics. Cognibotics is a direct spin-off from the RobotLab at the Faculty of Engineering (LTH) at Lund University and was founded in 2013. The company has since then broaden its work in several EU-funded projects as well as national scientific projects relating to automatic control, mechatronics and software technology.

Supported by Lund University Innovation System (LU Innovation), Vinnova and Lund University Holding company (LU Holding) Cognibotics has developed a range of commercial products and services. The team at Cognibotics shares an extensive network of contacts with robot experts in both academia and the automation industry, in Sweden and abroad. The company specializes in calibration and precise robot control.

1.4 Clarification regarding the terminology

The algorithm is designed in a symmetric manner to have both robot arms acting as the master with the other one following as the slave. However, the implementation of the second robot as a virtual robot limits it from being used as a master. As the real robot was the only one that could easily be interacted with, thus it became the master arm for the implementation and testing.

With this in mind, the master robot is thus named the real robot and the slave robot is named the virtual robot for clarification and consistency for the reader.

2

Background

The implementation of haptic feedback between a master and slave device can be made using different approaches. One approach is the use of passive control laws to enforce coordination between the two robot arms [Lee and Li, 2005]. Another approach is point-to-point kinematic mapping [Chen et al., 2007].

The work in this thesis is largely based on the previous work by Mahdi Ghazaei Ardakani, Martin Karlsson, Klas Nilsson, Anders Robertsson and Rolf Johansson in the article "Master-Slave Coordination Using Virtual Constraints for a Redundant Dual-Arm Haptic Interface" [Ghazaei Ardakani et al., 2018]. It uses a constraint-based approach to couple two nonlinear serial arms in task space on a dynamic level and will be described later in Chapter 3. The implementation in the article was done without the inclusion of an F/T sensor. This thesis seeks to expand upon the result in this article by including an F/T sensor.

3

Theory

3.1 Overview and clarification

This chapter will explain the different parts of the haptic control algorithm and how they are used. However before this there is one thing that must be clarified.

This work is largely based on the algorithm described in the article "Master-Slave Coordination Using Virtual Constraints for a Redundant Dual-Arm Haptic Interface" [Ghazaei Ardakani et al., 2018].

Therefore the theory that is explained in this chapter is the work of the already mentioned developers of the algorithm. It is simply expanded upon with some deeper explanation of concepts for the benefit of the readers of this work.

With the clarification of creation out of the way are the algorithms constructed using a flexible constraint-based approach to map between master and slave manipulators. It is based on coupling between two nonlinear serial arms in task space on a dynamical level.

The control principle is a generic approach to the mapping between a master and a slave manipulator while respecting existing multi-body dynamics. This is done by designing a regulator that derives the forces that are required to enforce the desired virtual kinematic constraints on robots that themselves are compensated for the gravitational forces.

How the virtual constraints are created are explained in Chapter 3.2. Chapter 3.3 will explain how the dynamics for the system are derived and integrated with the virtual constraints to create the control law. The calculation of the error is explained in Chapter 3.4. The proof of stability can be found in Chapter 3.5. Chapter 3.6 concerns poorly conditioned Γ matrices. The implementation of the regulator that makes the system invariant for all constant offsets can be found in Chapter 3.7. Chapter 3.8 explains the creation and implementation of the homogeneous transformation matrix, as well as the transformation of a geometric Jacobian into a body Jacobian.

Overarching plan

The dynamics of the manipulators can be represented by the following nonlinear system.

$$\begin{aligned} \dot{x} &= f(x) + g(x)u(x) \\ y &= h(x) \end{aligned} \tag{3.1}$$

where x denotes the state vector, u denotes the control signal and y denotes the system output. In this approach y is defined as the deviation of the translation and orientation of the fixed frame on the virtual robot arm from the fixed frame on the real robot arm plus the fixed desired offset. These frames can without loss of generality be located at the outer end of the robot, hereby referred to as the end-effector.

The goal of the regulator is to find $u(x)$ that results in y becoming identically equal to zero and to find the zero dynamics for the system. This is called zeroing the output by imposing a virtual constraint and will make sure that the end-effectors maintain the desired fixed offset.

Some concepts of robotics

Degrees of freedom (dof) Degrees of freedom (dof) is the smallest number of real-valued coordinates needed to represent the configuration of the robot. If the first arm of the two robots in this algorithm have n_1 dof and the second arm has n_2 dof then the total robotic system has $n = n_1 + n_2$ dof.

Configuration Configuration is the robot's position and it is a specification of all the points of the robot. As a robot is made up of rigid links of a known shape, only a few numbers are required and these numbers will be referred to as generalized coordinates. The generalized coordinates for the total robotic system in this algorithm are the joint angles in both arms.

Denoting the generalized coordinates for the first arm by $q_1 \in \mathbb{R}^{n_1}$ and the second arm by $q_2 \in \mathbb{R}^{n_2}$ allows us to denote the generalized coordinates for both arms by $q \in \mathbb{R}^n$ as $q = [q_1^T, q_2^T]^T$.

Rotation representation During this thesis rotation will be represented in three forms

- **Euler angles:** The Euler angles will be expressed in XYZ intrinsic angles. This will be used for visualization of the rotation error, as it is the most intuitive representation.
- **Rotation matrix:** $R \in SO(3)$ This will be used as the representation of rotation from real and virtual robots, as it is easy to work with.
- **Quaternions:** Quaternions are a way of representing rotation that only uses 4 numbers; a position axis (ε) and a rotation η around this axis. They will be used by the algorithm as they are more compact than the rotation matrix.

3.2 Virtual constraints

The virtual constraints used to create the control law can be divided into two parts, geometric and kinematic constraints. A geometric constraint is a relationship between parts of a geometric figure. A kinematic constraint is a constraint between rigid bodies that as a result decreases the degrees of freedom of the rigid body system. It is the kinematic constraint that will be implemented into the regulator, and it is derived from the geometric constraint.

The geometric constraint for this algorithm will be the difference in position and orientation between the two end-effectors of the robot arms. Denoting the position of the end-effector by $p_1 \in \mathbb{R}^3$ for the real robot arm and $p_2 \in \mathbb{R}^3$ for the virtual robot arm expressed in a common coordinate system as well as the orientation $R_1 \in SO(3)$ and $R_2 \in SO(3)$ for the respective arms allow us to express the geometric constraints between the two end-effectors.

$$p_2 - p_1 = \Delta p \tag{3.2a}$$

$$R_1^T R_2 = \Delta R \tag{3.2b}$$

where Δp denotes the positional offset and ΔR denotes the rotational offset. The kinematic constraint can be determined by differentiating the geometric constraints. Multiplying Equation (3.2b) by R_1 from the left and using the orthogonality condition ($R_1^T = R_1^{-1}$) [Lynch and Park, 2019a] results in $R_2 = R_1 \Delta R$. Differentiating both sides with regard to time results in

$$S(\omega_2)R_2 = S(\omega_1)R_1\Delta R + 0 = S(\omega_1)R_2 \Longleftrightarrow S(\omega_2 - \omega_1)R_2 = 0 \tag{3.3}$$

where $S(\omega)$ is the skew-symmetric matrix corresponding to the vector product by the angular velocity, $S(\omega_2 - \omega_1)R_2 = (\omega_2 - \omega_1) \times R_2$. Fixed relative positions and orientations imply that $R_2 \neq 0$ resulting in the kinematic constraints

$$v_2 - v_1 = 0 \tag{3.4a}$$

$$\omega_2 - \omega_1 = 0 \tag{3.4b}$$

where $v_i = \frac{dp_i}{dt}$ for robot i. The kinematic constraints have to be expressed in the generalized coordinates before they can be used in the algorithm. This is done by using $v = J_p(q)\dot{q}$ and $\omega = J_o(q)\dot{q}$ were $J_p(q)$ and $J_o(q)$ are the translational and rotational geometric Jacobians with respect to the end-effector, $\dot{q}$ is the joint velocities for each robot. These Jacobians are unique to their specific arm, resulting in

$$\begin{aligned} J_{2p}(q_2)\dot{q_2} - J_{1p}(q_1)\dot{q_1} = 0 \\ J_{2o}(q_2)\dot{q_2} - J_{1o}(q_1)\dot{q_1} = 0 \end{aligned} \tag{3.5}$$

Rewriting Equation (3.5) with

$$G = \begin{pmatrix} -J_{1p}(q_1) & J_{2p}(q_2) \\ -J_{1o}(q_1) & J_{2o}(q_2) \end{pmatrix} = [-J_1, J_2] \in \mathbb{R}^{6\times n} \tag{3.6}$$

results in the kinematic constraints

$$G\dot{q} = 0 \tag{3.7}$$

3.3 Dynamic Coupling

We need to calculate the equation of motion to derive the dynamics for the system. This will be done by the use of the Lagrangian formulation, with the Lagrangian function $\mathscr{L}(\dot{q},q)$ [Lynch and Park, 2019b]. The Lagrangian function is calculated as the kinetic energy $\mathscr{K}(\dot{q},q)$ minus the potential energy $\mathscr{P}(,q)$.

$$\mathscr{L}(\dot{q},q) = \mathscr{K}(\dot{q},q) - \mathscr{P}(,q) \tag{3.8}$$

The equation of motion is then calculated from the Lagrangian function according to

$$\tau = \frac{d}{dt}\frac{\delta \mathscr{L}}{\delta \dot{q}} - \frac{\delta \mathscr{L}}{\delta q} \tag{3.9}$$

However, the goal of the regulator is not to compensate for the gravitational forces as stated in the overview resulting in that from the regulator's perspective there is no potential energy to be included in the Lagrangian function. This results in

$$\mathscr{L}(\dot{q},q) = \mathscr{K}(\dot{q},q) \tag{3.10}$$

The kinetic energy of the rigid body can be expressed as the sum of its linear and rotational velocity components according to

$$\mathscr{K} = \frac{1}{2}mv^T v + \frac{1}{2}\omega^T I\omega \tag{3.11}$$

where m is the mass of the rigid body, v and ω is the linear and rotational velocity for the rigid body with I is the inertia matrix for the rigid body. Equation (3.11) expanded for all i links in the robot arm and expressed in the generalized coordinates results in the following expression for the total kinetic energy of the robot arm.

$$\mathscr{K} = \frac{1}{2}\sum_i m_i \dot{q}^T (J_p^i)^T J_p^i \dot{q} + \frac{1}{2}\sum_i \dot{q}^T (J_o^i)^T R_{l_i} I_i R_{l_i}^T J_o^i \dot{q} \tag{3.12}$$

In Equation (3.12) m_i and I_i is the mass and inertia matrix for the link i. J_p^i and J_o^i denote the partial and rotational Jacobians for the link i and R_{l_i} is the rotational matrix of link i expressed in the base coordinate system.

The result of the equation of motion calculated from Equation (3.9) is often restructured in the following way by gathering terms with similar makeup into separate parts.

$$\tau = M(q)\ddot{q} + C(\dot{q},q) \tag{3.13}$$

In Equation (3.13) $M(q)$ is the symmetric positive-definite mass matrix, $C(\dot{q},q)$ is the vector containing the Coriolis and centripetal torques and $\ddot{q}$ is the joint accelerations. In addition there will also be losses due to friction, modelled using $-\mu_v\dot{q}$ under the assumption that there is only viscous friction with coefficient μ_v. There will also be a torque from generalized external forces Q^e and the torque from generalized forces due to the kinematic constraints are denoted by Q^{kc}. The Lagrange–d'Alembert theorem states that

"The sum of the differences between the forces acting on a system of massive particles and the time derivatives of the momenta of the system itself projected onto any virtual displacement consistent with the constraints of the system is zero." [Bloch, 2003].

This included in the complete equation of motion results in

$$\begin{aligned} &M(q)\ddot{q}+C(\dot{q},q)\dot{q}=Q^e+Q^{kc}-\mu_v\dot{q} && Q^e=\tau+J_b^T h^e \\ &G\dot{q}+g_0=0 && Q^{kc}=G^T\lambda \end{aligned} \tag{3.14}$$

In Equation (3.14) the generalized external forces Q^e have two components, the external torque applied at the joints τ and the torque that results from the forces and torques on the end effector h^e multiplied with the body Jacobian J_b. The generalized forces due to the kinematic constraints Q^{kc} are calculated as the Lagrange multipliers $\lambda(t,\dot{q},q)\in\mathbb{R}^6$ multiplied with G.

Equation (3.14) has terms that depend on information from one arm and terms that depend on information from both arm. Splitting up Equation (3.14) by the use of subscripts 1 and 2 for the parameters and variables, we find that

$$\begin{aligned} &M(q) := blkdiag(B_1(q_1),B_2(q_2)) \\ &J(q) := blkdiag(J_1(q_1),J_2(q_2)) \\ &J_b(q) := blkdiag(J_{b1}(q_1),J_{b2}(q_2)) \\ &C(\dot{q},q) := blkdiag(C_1(\dot{q}_1,q_1),C_2(\dot{q}_2,q_2)) \\ &h^e := \begin{pmatrix} h_1^e \\ h_2^e \end{pmatrix} \quad Q^{kc} := \begin{pmatrix} Q_1^{kc} \\ Q_2^{kc} \end{pmatrix} = \begin{pmatrix} -J_1^T(q_1)\lambda \\ J_2^T(q_2)\lambda \end{pmatrix} \end{aligned} \tag{3.15}$$

resulting in the following equation of motion for each individual arm

$$B_i(q_i)\ddot{q}_i+C_i(\dot{q}_i,q_i)\dot{q}_i=\tau_i+J_{bi}^T(q_i)h_i^e+Q_i^{kc}-\mu_v\dot{q}_i \tag{3.16}$$

Equation (3.15 and 3.16) $B_i(q_i)$ is the mass matrix of the arm i, $C_i(\dot{q}_i,q_i)$ is the Coriolis and centripetal matrix, $J_i(q_i)$ is the geometric Jacobian and $J_{bi}(q_i)$ is the body Jacobian for arm i.

The last thing to calculate is the Lagrange multipliers $\lambda(t,\dot{q},q)$ that will be the control signal u. By introducing $x^T=(q^T,\dot{q}^T)$ the complete equation of motion in

Equation (3.14) can be rewritten as.

$$\dot{x} = \begin{pmatrix} \dot{q} \\ M^{-1}(q)(-C(q,\dot{q})\dot{q} - \mu_v\dot{q} + Q^e + G^T(q)u) \end{pmatrix} =: f(x) + g(x)u \tag{3.17a}$$

$$\dot{y} = G(q)\dot{q} \tag{3.17b}$$

where $M^{-1}(q) = blkdiag(B_1^{-1}(q_1), B_2^{-1}(q_2))$.

Equation (3.14) defines a differential-algebraic equation system, which can be solved numerically. This solution is the zero dynamics of the system in Equation (3.17a) fulfilling the control principle set up in the overview sections.

By defining $\Gamma := GM^{-1}G^T$ the control law $u^*(x)$ that makes the system invariant for all constant offsets becomes

$$u^* = \Gamma^{-1}(GM^{-1}(C\dot{q} - \tau - J_b^T h^e + \mu_v\dot{q}) - \dot{G}\dot{q}) \tag{3.18}$$

where

$$\dot{G} := \sum_{k=1}^{n} \left(\frac{\delta G}{\delta q_k}\dot{q}_k \right) = [-\dot{J}_1, \dot{J}_2] \tag{3.19}$$

The complete calculations of u^* in (3.18) can be found in [Ghazaei Ardakani et al., 2018]. The zero dynamics for Equation (3.17a) are given by

$$M(q)\ddot{q} + W(C(q,\dot{q})\dot{q} - Q^e + \mu_v\dot{q}) + G^T\Gamma^{-1}\dot{G}\dot{q} = 0 \tag{3.20}$$

where $W = I_{n\times n} - P$ and $P = G^T\Gamma^{-1}GM^{-1}$.

Under the assumption that the block-diagonal matrices $K_p = blkdiag(K_{tp}, K_{op})$ and $K_d = blkdiag(K_{td}, K_{od})$ are positive definite. The state variable feedback

$$\lambda = u = u^* - \Gamma^{-1}(K_d\dot{y} + K_p e) \tag{3.21}$$

results in the asymptotic stability of Equation (3.17a) where e is the translational and rotational error between the end-effectors of the robots. K_p is the proportional constant and K_d is the derivative constant that is subdivided dependent if they act on the transitional or orientation part. This results in

- K_{tp} is the transitional proportional constant
- K_{op} is the orientation proportional constant
- K_{td} is the transitional derivative constant
- K_{od} is the orientation derivative constant

The proof of stability will be expanded upon in Chapter 3.5.

The state variable feedback created in Equation (3.21) can be seen as a virtual force that affects the robots and this force is transformed to the torque that is to be applied on the individual joints as

$$\tau_{joint} = Q^{kc} = G^T u \tag{3.22}$$

3.4 Calculation of error

The calculation of the error can be distinctly divided into two parts; translational (e_p) and rotational (e_o) error.

$$e = \begin{pmatrix} e_p \\ e_o \end{pmatrix} \tag{3.23}$$

The translational error is calculated as the difference between the positions of the two end-effectors with the chosen positional offset Δp.

$$e_p = p_2 - (p_1 + \Delta p) \tag{3.24}$$

The rotational error is calculated as the difference between the rotation of the two end-effectors with the chosen rotational offset ΔR.

$$R_e = R_2(R_1 \Delta R)^T \tag{3.25}$$

The algorithm needs to express the rotational error (R_e) as an $\mathbb{R}^3$ vector. This is done by expressing the rotational error (e_o) as the position axis (ε) of the quaternion, with the rotation η used to scale the error when used by the orientation proportional constant (K_{op}).

The transformation from rotation matrix to quaternion is done through [Lynch and Park, 2019c]

$$\begin{aligned} R_e &= \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \\ \eta &= \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}} \\ \varepsilon &= \frac{1}{4\eta} \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix} \end{aligned} \tag{3.26}$$

resulting in

$$e = \begin{pmatrix} e_p \\ e_o \end{pmatrix} = \begin{pmatrix} p_2 - (p_1 + \Delta p) \\ 2\frac{\varepsilon}{sin(\eta/2)} \end{pmatrix} \tag{3.27}$$

3.5 Proof of stability

Inserting Equation (3.21) into Equation (3.17a) results in

$$\ddot{y} + K_d\dot{y} + K_p e = 0 \tag{3.28}$$

Expanding Equation (3.20) with the definition of K_p and K_d results in

$$\ddot{e}_p + K_{td}\dot{e}_p + K_{tp}e_p = 0 \tag{3.29a}$$
$$\dot{\tilde{\omega}} + K_{od}\tilde{\omega} + K_{op}e_o = 0 \tag{3.29b}$$

where $\tilde{\omega} = \omega_2 - \omega_1$. The translational part in Equation (3.29a) is a linear system and exponential stable as K_{td} and K_{tp} are positive definite.

Proving that the rotational part in Equation (3.29b) is stable can be done by LaSalle's theorem.

Let $V(x)$ be a Lyapunov function for $\dot{x} = f(x)$ in the set $x \in \Omega$ such that $\dot{V} \leq 0$ for $x \in \Omega$. Let E be a set of all points in Ω where $\dot{V}(x) = 0$ and set M to be the largest invariant set in E. Then every solution starting in Ω approaches M as $t \longrightarrow \infty$. [Johansson, 2019]

The first step in using LaSalle's theorem is the creation of a Lyapunov candidate function.

$$V = \frac{1}{2}\tilde{\omega}^T K_{op}^{-1}\tilde{\omega} + (\eta_2 - \eta_1)^2 + (\varepsilon_2 - \varepsilon_1)^T(\varepsilon_2 - \varepsilon_1) \tag{3.30}$$

where η_i is the angle and ε_i is the unit vector of the unit quaternion for arm i. Taking the time derivative along the system trajectory of Equation (3.30) results in

$$\dot{V} = -\tilde{\omega}^T K_{op}^{-1} K_{od}\tilde{\omega} \leq 0 \tag{3.31}$$

concluding that $e_o = 0$ and $\tilde{\omega} = 0$ is globally asymptotically stable. As K_d and K_p are positive definite the solution of Equation (3.28) converges asymptotically to zero.

3.6 Ill-conditioned Γ and the damped pseudo-inverse

In a few configurations G becomes rank deficient resulting in Γ becoming ill-conditioned. This problem, though rare, as this would require that the rank of G falls below 6 can be solved by replacing Γ^{-1} in Equation (3.21) with its damped pseudo-inverse [Burdick, 2020].

$$\Gamma_\rho^{-1} = \Gamma^T(\Gamma\Gamma^T + \rho^2 I)^{-1} \tag{3.32}$$

where $\rho << 1$ is the dampening factor.

3.7 Implementation of u^*

The implementation of the algorithm was slightly changed from its implementation in the previous work [Ghazaei Ardakani et al., 2018] and the change is in the implementation of $u^*(x)$ in Equation (3.18).

$$u^* = \Gamma^{-1}(GM^{-1}(C\dot{q} - \tau - J_b^T h^e + \mu_v \dot{q}) - \dot{G}\dot{q})$$

where τ is the torque applied directly to the joints of the robot from outside sources. Including this information will help predict the movements of the robotic arm when the robot is moved with other means than interaction with the handle. This feature, however, is not implemented as a result of limited time. This change results in the following equation for the calculation of the control signal $u^*(x)$ that makes the system invariant for all constant offsets

$$u^* = \Gamma^{-1}(GM^{-1}(C\dot{q} - J_b^T h^e + \mu_v \dot{q}) - \dot{G}\dot{q}) \tag{3.33}$$

Equation (3.33) was then expanded using Equation (3.15) into

$$\begin{aligned}
u^* &= \Gamma^{-1}\Big(\begin{pmatrix} -J_1 & J_2 \end{pmatrix} \begin{pmatrix} B_1^{-1} & 0 \\ 0 & B_2^{-1} \end{pmatrix} \Big(\begin{pmatrix} C_1 & 0 \\ 0 & C_2 \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} - \begin{pmatrix} J_{b1}^T & 0 \\ 0 & J_{b2}^T \end{pmatrix} \begin{pmatrix} h_1^e \\ h_2^e \end{pmatrix} \\
&+ \begin{pmatrix} \mu_1 \dot{q}_1 \\ \mu_2 \dot{q}_2 \end{pmatrix} \Big) - \begin{pmatrix} -\dot{J}_1 & \dot{J}_2 \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} \Big) = \\
&\Gamma^{-1}\Big(\begin{pmatrix} -J_1 B_1^{-1} & J_2 B_2^{-1} \end{pmatrix} \Big(\begin{pmatrix} C_1 \dot{q}_1 \\ C_2 \dot{q}_2 \end{pmatrix} - \begin{pmatrix} Jb_1^T h_1^e \\ Jb_2^T h_2^e \end{pmatrix} + \begin{pmatrix} \mu_1 \dot{q}_1 \\ \mu_2 \dot{q}_2 \end{pmatrix} \Big) - (-\dot{J}_1 \dot{q}_1 + \dot{J}_2 \dot{q}_2) \Big) = \\
&\Gamma^{-1}\Big(-J_1 B_1^{-1}(C_1 \dot{q}_1 - J_{b1}^T h_1^e + \mu_1 \dot{q}_1) + J_2 B_2^{-1}(C_2 \dot{q}_2 - J_{b2}^T h_2^e + \mu_2 \dot{q}_2) \\
&- (-\dot{J}_1 \dot{q}_1 + \dot{J}_2 \dot{q}_2) \Big) \\
&= \Gamma^{-1}(-A_1 + A_2), \quad \textit{where } A_i = J_i B_i^{-1}(C_i \dot{q}_i - J_{bi}^T h_i^e + \mu_i \dot{q}_i) - \dot{J}_i \dot{q}_i
\end{aligned} \tag{3.34}$$

$u^*(x)$ in Equation (3.34) was implemented into Python in three parts. These three parts are Γ^{-1}, A_1 and A_2.

- Γ^{-1} that requires information of the joint angles from both robots.
- A_1 that requires information from the real robot in the form of joint angles q_1, joint angular velocities $\dot{q}_1$ and the forces and torques that are applied to the end effector h_1^e. μ_1 is the viscous friction constants for the real robot.
- A_2 that requires information from the virtual robot in joint angles q_2, joint angular velocities $\dot{q}_2$ and the forces and torques that are applied to the end effector h_2^e. μ_2 is the viscous friction constants for the virtual robot.

This change was implemented for two reasons

- To divide the algorithm into clear parts dependent on each robot.
- Provide departmentalization for ease of troubleshooting.

3.8 Homogeneous transformation matrices and the types of Jacobians

Homogeneous transformation matrices

The homogeneous transformation matrix is one way of representing the combined orientation and position of a rigid body. It is made up of a rotational matrix $R \in SO(3)$ to represent the orientation of the body-frame $\{b\}$ in the fixed base frame $\{s\}$ and a vector $p \in \mathbb{R}^3$ that represents the origin of $\{b\}$ in $\{s\}$. They are combined according to Equation (3.35) [Lynch and Park, 2019d].

$$T = \begin{pmatrix} R & p \\ 0 & 1 \end{pmatrix} \tag{3.35}$$

This makes it easy to go from one frame to another. To give structure to the different frames, the fixed base frame is referred to as $\{0\}$ and each following frame is given a rising number reaching the frame of the flange of the robot's end-effector $\{7\}$. An overview of the frames can be seen in Figure D.1 in Appendix D.

To go from the fixed base frame $\{0\}$ to the frame of the flange of the robot's end-effector $\{7\}$ is done by

$$T_{07} = T_{01} \cdot T_{12} \cdot T_{23} \cdot T_{34} \cdot T_{45} \cdot T_{56} \cdot T_{67} \tag{3.36}$$

The homogeneous transformation matrices used in the calculation of the transformation matrix from the fixed space frame to the body frame at the end of the end effector flange can be seen in Equations (3.37-3.43) and are based on the frames seen in Figure D.1 in Appendix D.

$$T_{01} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.37}$$

$$T_{12} = \begin{pmatrix} cos(q_1) & 0 & sin(q_1) & 0.069 \cdot sin(q_1) \\ sin(q_1) & 0 & -cos(q_1) & -0.069 \cdot cos(q_1) \\ 0 & 1 & 0 & 0.063 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.38}$$

$$T_{23} = \begin{pmatrix} cos(q_2) & -sin(q_2) & 0 & -0.425 \cdot cos(q_2) \\ sin(q_2) & cos(q_2) & 0 & -0.425 \cdot sin(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.39}$$

$$T_{34} = \begin{pmatrix} cos(q_3) & -sin(q_3) & 0 & -0.392 \cdot cos(q_3) \\ sin(q_3) & cos(q_3) & 0 & -0.392 \cdot sin(q_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.40}$$

$$T_{45} = \begin{pmatrix} cos(q_4) & 0 & sin(q_4) & 0.05 \cdot sin(q_4) \\ sin(q_4) & 0 & -cos(q_4) & -0.05 \cdot cos(q_4) \\ 0 & 1 & 0 & 0.065 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.41}$$

$$T_{56} = \begin{pmatrix} cos(q_5) & 0 & -sin(q_5) & 0.051 \cdot sin(q_5) \\ sin(q_5) & 0 & cos(q_5) & -0.051 \cdot cos(q_5) \\ 0 & -1 & 0 & 0.05 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.42}$$

$$T_{67} = \begin{pmatrix} cos(q_6) & -sin(q_6) & 0 & 0 \\ sin(q_6) & cos(q_6) & 0 & 0 \\ 0 & 0 & 1 & 0.049 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.43}$$

Geometric and body Jacobian

The Jacobian $J(\theta) \in R^{6\times n}$ is a matrix that relates the impact of forces, torques and velocities on the end effector to torque and joint angular velocities. The Jacobian is dependent on the n joint angles θ of the robot.

Jacobians are differentiated based on in what frame of reference the forces, torques and velocities that affect the end effector are expressed in. The two types that will be important for this thesis are

- **The geometric Jacobian:** $J_s(\theta) \in R^{6\times n}$ relates the joint velocity vector $\dot{\theta} \in R^n$ to the end effector linear and angular velocity V_s that is expressed in the base frame $\{s\}$ via $V_s = J_s(\theta)\dot{\theta}$.
- **The body Jacobian:** $J_b(\theta) \in R^{6\times n}$ relates the joint velocity vector $\dot{\theta} \in R^n$ to the end effector linear and angular velocity V_b that is expressed in the body frame $\{b\}$ via $V_b = J_b(\theta)\dot{\theta}$.

The base frame $\{s\}$ and body frame $\{b\}$ are graphically presented in Figure 3.1. There is a third type that needs to be mentioned and that is the tool Jacobian.

- **The tool Jacobian:** $J_t(\theta) \in R^{6\times n}$ relates the joint rate vector $\dot{\theta} \in R^n$ to the end effector linear and angular velocity V_t that is expressed in the tool frame $\{t\}$ via $V_t = J_t(\theta)\dot{\theta}$.

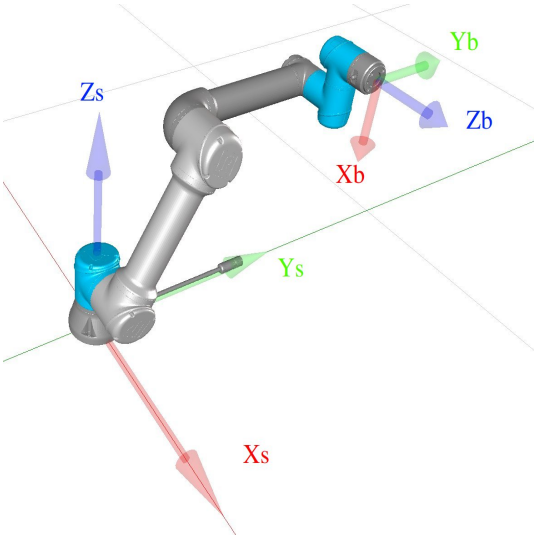


Figure 3.1 The coordinate axis of the base frame $\{s\}$ and the coordinate axis for the body frame $\{b\}$.

The difference between the tool and body Jacobian is that the tool Jacobian takes the relative transformation between of the F/T sensor attached to the end effector into account. The difference between tool and body Jacobian in this particular implementation was so small that the body Jacobian was used to minimizing the calculation time.

The transformation between the geometric and body Jacobian is done by rotating the old frame to the new frame using a rotation matrix [Siciliano et al., 2009]. Transform from $\{s\}$ to $\{b\}$ is done through

$$J_b = \begin{pmatrix} R_{bs} & 0 \\ 0 & R_{bs} \end{pmatrix} J_s \tag{3.44}$$

resulting in

$$J_7 = \begin{pmatrix} R_{70} & 0 \\ 0 & R_{70} \end{pmatrix} J_0 \xrightarrow{R_{70}=R_{07}^{-1}=R_{07}^T} J_7 = \begin{pmatrix} R_{07}^T & 0 \\ 0 & R_{07}^T \end{pmatrix} J_0 \tag{3.45}$$

where R_{07}^T is the transpose of the rotational part of the homogeneous transformation matrix T_{07} in Equation (3.36).

Transferring forces and torques between frames

Forces and torques are transferred between frames similarly to the transformation between different Jacobians with the lever interaction taken into account by the use of adjoint representation.

Definition Given $T = (R, p) \in SE(3)$ as a homogeneous transformation matrix, then the adjoint representation $[Ad_T]$ [Lynch and Park, 2019e] is

$$[Ad_T] \hat{=} \begin{pmatrix} R & 0 \\ [p]R & R \end{pmatrix} \in \mathbb{R}^{6\times 6} \quad p = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \longrightarrow [p] = \begin{pmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{pmatrix} \tag{3.46}$$

This definition is, however, not entirely applicable as it is defined for the following definition of a twist. A twist ($\mathscr{V}$) is a combination of the three-dimensional quantities angular (ω) and a linear (v) velocity, it is used to describe a motion in a compact single six-dimensional vector.

$$\mathscr{V} = \begin{pmatrix} \omega \\ v \end{pmatrix}$$

This definition is for changing the frame of reference for the spiral twists according to

$$\begin{pmatrix} \omega_s \\ v_s \end{pmatrix} = \begin{pmatrix} R & 0 \\ [p]R & R \end{pmatrix} \begin{pmatrix} \omega_b \\ v_b \end{pmatrix} \tag{3.47}$$

As we in this thesis use the notation with the linear velocity v_b on top of the angular velocity ω_s we will be rewriting it as

$$\begin{pmatrix} v_s \\ \omega_s \end{pmatrix} = \begin{pmatrix} R & [p]R \\ 0 & R \end{pmatrix} \begin{pmatrix} v_b \\ \omega_b \end{pmatrix} \longrightarrow [Ad_T] \hat{=} \begin{pmatrix} R & [p]R \\ 0 & R \end{pmatrix} \in \mathbb{R}^{6\times 6} \tag{3.48}$$

3.9 Summary

The dynamics of the system were calculated, introducing $x^T = (q^T, \dot{q}^T)$ resulted in the following equation of motion shown in Equation (3.17)

$$\begin{aligned} \dot{x} &= \begin{pmatrix} \dot{q} \\ M^{-1}(q)(-C(q,\dot{q})\dot{q} - \mu_v \dot{q} + \tau + J_b^T h^e + G^T(q)u) \end{pmatrix} =: f(x) + g(x)u \\ \dot{y} &= G(q)\dot{q} \end{aligned}$$

where

$$M^{-1}(q) := blkdiag(B_1^{-1}(q_1), B_2^{-1}(q_2))$$
$$J_b(q) := blkdiag(J_{b1}(q_1), J_{b2}(q_2))$$
$$C(\dot{q}, q) := blkdiag(C_1(\dot{q}_1, q_1), C_2(\dot{q}_2, q_2))$$
$$h^e := \begin{pmatrix} h_1^e \\ h_2^e \end{pmatrix}$$
$$\tau := \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix}$$

Additionally, $B_i^{-1}(q_i), J_{bi}(q_i), C_i(\dot{q}_i, q_i), h_i^e$ and τ_i are the inverted mass matrix, body Jacobean, Coriolis, external forces and torque applied to the end effector matrices and the external torque applied to the joints for arm i.

From the equation of motion the control principle is constructed, resulting in the control signal $u^*(x)$ that makes the system invariant for all constant offsets seen in Equation (3.18)

$$u^* = \Gamma^{-1}(GM^{-1}(C\dot{q} - \tau - J^T h^e + \mu_v \dot{q}) - \dot{G}\dot{q})$$

where $\Gamma := GM^{-1}G^T$ and $\dot{G} = [-\dot{J}_1, \dot{J}_2]$. The state variable feedback calculated in Equation (3.21) then becomes

$$\lambda = u = u^* - \Gamma^{-1}(K_d \dot{y} + K_p e)$$

and is used to achieve asymptotic stability with the assumption that the block-diagonal matrices $K_p = blkdiag(K_{tp}, K_{op})$ and $K_d = blkdiag(K_{td}, K_{od})$ are positive definite. e is the translational and rotational error.

The implementation of the control signal is then modified to better fit as

$$u^* = \Gamma^{-1}(-A_1 + A_2), \quad where\ A_i = J_i B_i^{-1}(C_i \dot{q}_i - J_{bi}^T h_i^e + \mu_i \dot{q}_i) - \dot{J}_i \dot{q}_i$$

resulting in the control signal becoming

$$u = u^* - \Gamma^{-1}(K_d \dot{y} + K_p e) = \Gamma^{-1}(-A_1 + A_2) - \Gamma^{-1}(K_d \dot{y} + K_p e)$$

The control signal is implemented as virtual forces that affect the robot's. The final step is then to transform these virtual forces into torques that are applied on the individual joints. This is calculated as

$$\tau_{joint} = G^T u$$

4

Modelling

The robot used in the project is the UR5e robot arm. It is a medium sized robot designed by Universal Robots. It has 6 dof, a payload capacity of 5 kg and a reach of 850 mm.

At first was a model of a UR5e robot was made in the simulation software Maplesim [*Maplesim* 2020]. This was so that models of the robots and the controller could be tested in a simulated environment with the purpose of getting acquainted with the algorithm. The second reason for implementation into Maplesim was that the simulation of the robot would serve as as the slave robot by running the created simulation of the virtual robot in parallel with the real robot.

These reasons would gradually become irrelevant as the project progressed and the limitations of the simulation software were made apparent. These reasons will be discussed further in Chapter 4.2.

With the limitations of the simulation software preventing a full implementation of the algorithm into Maplesim, was the focus shifted to creating the model for the robot in Maplesim and then implement its model into Python with the rest of the controller. However, further limitations would reduce the usefulness of the Maplesim model to the point that it was replaced with a simpler model implemented in Python code.

4.1 Creation of the UR5e model in Maplesim

The model of the UR5e robot arm was made in Maplesim 2020 using Denavit-Hartenberg parameters from the drawing of the UR5e robot publicly available from the website shown in Appendix A, Figure A.1. This was done as this robot was envisioned to have its mass, Coriolis and Jacobian matrix calculated by the Cognibotics matrix calculation code in `cdyn` library. This required that the robot has the same kinematics as the model inside the company's matrix calculation code. These changes amounted to rounding of the Denavit-Hartenberg parameters provided by Universal robots [*Denavit–Hartenberg parameters* 2020]. The difference between these two sets of kinematic dimensions can be seen in Table 4.1. There were no

Joint	a[m] UR	a[m] CB	d [m] UR	d [m] CB	α[rad] UR	α[rad] CB
1	0	0	0.1625	0.163	$\pi/2$	$\pi/2$
2	-0.425	-0.425	0	0	0	0
3	-0.3922	-0.392	0	0	0	0
4	0	0	0.1333	0.131	$\pi/2$	$\pi/2$
5	0	0	0.0997	0.1	$-\pi/2$	$-\pi/2$
6	0	0	0.0996	0.1	0	0

Table 4.1 Difference in Denavit-Hartenberg parameters between the one provided by Universal robot's parameters (UR) and the one used by the Cognibotics matrix calculation code (CB).

dynamic parameters given for the older version of the kinematics, so the dynamic parameters provided by Universal robots [*Denavit–Hartenberg parameters* 2020] were used. As the difference between them in kinematic parameters was only a rounding into millimetres the dynamic parameters were considered close enough to be used.

Each joint were created as a subsystem and then connected. A crude CAD drawing was divided into seven parts and connected to each joint including the base for a better visual representation.

The robot was then divided into three parts, represented as three subsystems.

- **Environment part:** The environment part contains an FT sensor connected to a spring that in turn is connected to the ground, which is done to simulate impact with the environment. This part can be seen in Appendix B, Figure B.2.
- **Drive part:** This is where the input signal to the robot is used as torque for the joints. Joint angle and joint velocity are also placed here. This part can be seen in Appendix B, Figure B.3.
- **Mechanical part:** This is where the six Denavit-Hartenberg subsystems are connected. This part is connected to the drive part using the axis and to the environment part by its end effector. This part can be seen in Appendix B, Figure B.4.

Drawings of the different parts, as well as the connected system can be seen in Appendix B Figures B.1 - B.4.

Finally, the three parts were all included into one subsystem for ease of converting into a format that Python could interpret.

4.2 Implementation of the algorithm in Maplesim

The idea of implementing the algorithm into Maplesim using two simulated UR5e robots as inputs and outputs proved complicated as Maplesim did not have a block dedicated to matrix creation or matrix multiplication.

The result was that the implementation of the algorithm became a large block without any clear overview. This compiled with difficulty to calibrate the control parameters resulted in an abandonment of the attempt to simulate the algorithm with the regulator.

The exercises, however, provided insights into the complexity of the algorithm as well as its dependence on a time-efficient way of calculating the mass, Coriolis and Jacobian matrix for both robots.

4.3 Creation of the virtual robot from Maplesim model

The next step after creating a model for the virtual robot created in Maplesim is to implement this model into the Python. There are a few concepts that have to explained before describing the implementation of the Maplesim model as a virtual robot in Python.

- **Functional Mock-up Unit (FMU):** An FMU is a file that contains a simulation of a model adherent to the widely used Functional Mock-up Interface. This allows one to save a simulation in a way that is usable by most simulation and code programs. There are two different versions of FMU and although both are dynamical systems represented by differential equations are the way the solvers implemented is different. The equations are the same but how they are solved differ between the two versions [*FMU* 2020].
  - **Model Exchange Functional Mock-Up Units (ME FMU):** The numerical solver in a ME FMU is supplied by the importing tool. This means that the equations are solved using the algorithm that the importer provides.
  - **Co Simulation Functional Mock-Up Units (CS FMU):** The numerical solver in a CS FMU is supplied by the exporting tool. This means that the equations as well as the algorithm used to solve them are provided by the exporter.
- **PyFMI:** PyFMI is a package for loading and interacting with FMUs in Python [*PyFMI* 2020].

The Maplesim model was used for the creation of the virtual robot by exporting it as an ME FMU. ME was chosen primarily for ease of use, CS would provide a better simulation as CS allows the use of Maplesim's more capable numerical solver.

There were serious, to this day, unresolved problems with the CS FMU not being correctly interpreted by PyFMI. Lengthy correspondence with the company that created Maplesim, Modelon were not able to solve the problem. For this reason the ME FMU option was chosen over the CS FMU option.

This, however, was not the largest problem threatening the use of ME FMU as the core of the virtual robot. In order to use the ME FMU model as the virtual robot the mass and dynamic parameters needed to be identified. This is to allow Cognibotics matrix calculation code to construct its mass, Coriolis and Jacobian matrices.

The identification software was meant to be use on real robots and required robot's joint angles and motor currents to identify the robot's parameters. The current was simulated by taking the joint torque and using it to calculate the motor current. Despite this workaround it was hard to identify the mass and dynamic parameters for the Maplesim model.

4.4 Creation of the virtual robot from inverse dynamics

Another simpler way of simulating the virtual robot was used as a response to the problems with the FMU implementation into Python. This simpler model required the input of joint torque and the outputs of joint angles and joint velocities.

This is done in two steps, first the joint acceleration is calculated using the expression for the inverse dynamics

$$\begin{aligned}
&\tau = M(\theta)\ddot{\theta} + C(\theta,\dot{\theta}) + g(\theta) - \mu\dot{\theta} - J_b^T(\theta)h_e \Longleftrightarrow \\
&\ddot{\theta} = M(\theta)^{-1}(\tau - C(\theta,\dot{\theta}) - g(\theta) + \mu\dot{\theta} + J_b^T(\theta)h_e) \xrightarrow{H(\theta,\dot{\theta}) \hat{=} C(\theta,\dot{\theta}) + g(\theta)} \\
&\ddot{\theta} = M(\theta)^{-1}(\tau - H(\theta,\dot{\theta}) + \mu\dot{\theta} + J_b^T(\theta)h_e)
\end{aligned} \tag{4.1}$$

were $M(\theta)$, $C(\theta,\dot{\theta})$, are the mass, Coriolis matrices and $g(\theta)$ is the gravity vector for the virtual robot respectively, μ is the viscus friction for the virtual robot and θ, $\dot{\theta}$ and $\ddot{\theta}$ are joint angles, velocities and accelerations, h_e is the forces and torques that act on the robot's end-effector.

Second, given the joint acceleration from Equation (4.1) the joint angle and velocity were calculated using zero-order hold double integration. There the next joint angle (θ_{k+1}) and joint velocity ($\dot{\theta}_{k+1}$) is calculated from the current joint angle (θ_k), joint velocity ($\dot{\theta}_k$) and joint acceleration ($\ddot{\theta}_k$) as seen in Equation (4.2) [Åström and Wittenmark, 2011].

$$\begin{pmatrix} \theta_{k+1} \\ \dot{\theta}_{k+1} \end{pmatrix} = \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_k \\ \dot{\theta}_k \end{pmatrix} + \begin{pmatrix} \frac{h^2}{2} \\ h \end{pmatrix} \ddot{\theta}_k \tag{4.2}$$

This implementation solved one of the more time-consuming identification aspects of the virtual robot, the identification of the mass and dynamic parameter of the

virtual robot. By using the mass and dynamic parameter identified for the real robot and then creating the mass and Coriolis matrices for the virtual robot with Cognibotics code, the mass and Coriolis matrices for the virtual robot in Equation (4.1) the same as the ones used in Equation (3.34). This removes the need for identification of the mass and dynamic parameter of the virtual robot.

The mass and dynamic parameters used for the virtual were chosen to be the same as the identified mass and dynamic parameters for the real UR5e robot.

Creation of a virtual wall

A virtual wall was implemented to test the haptic response when the virtual robot collides with an object. This was implemented as a simple spring force designed to provide a repulsive force out of the virtual wall when the end effector from the virtual robot moved through it, according to Hooke's law:

$$F = -k\delta_p \tag{4.3}$$

The force F was calculated with the spring constant $k = 6000\ N/m$ and δ_p being the distance travelled into the wall. The magnitude of the spring constant was chosen as it resulted in an acceptable distance that the virtual robot was allowed into the virtual wall of 1-2 mm at maximum force applied to the real robots end effector. There is a limitation for large spring constants as a high stiffens of the virtual plane can result in instability if the sampling frequency is not high enough to handle the rapid increase in the repulsive force. This instability was however not experienced.

The wall was set up at $x = 0.45\ m$ with the normal vector of the wall pointing in the negative x-direction in the base coordinate frame. This force was then transformed from the base frame to the sensor frame using the adjoint representation in Equation (3.48).

4.5 Summary

The algorithm and a virtual duplicate of an UR5e robot were implemented in the simulation software Maplesim for testing and educational purposes. The simulation of the algorithm was abandoned due to the limitations in Maplesim referenced in chapter 4.2.

The Maplesim model of the UR5e was implemented as the virtual robot arm when the algorithm was implemented into Python code. However, problems with identification, bad integration, less than ideal solver and shortage of time limited its use.

The model for the virtual robot was simplified and instead based on inverse dynamics for the calculation of the joint acceleration and double integration for the approximation of joint angle and joint velocity.

A virtual wall was implemented to test the haptic response when the virtual robot collide with an object as a spring force at $x = 0.45\ m$ with the normal vector of the wall pointing in the negative x-direction in the base coordinate frame.

5

Experimental Setup

5.1 General overview

The experimental setup consists of four parts. The UR5e robot, the handle, the F/T sensor and a computer on which the regulator and the virtual robot together with other processes run. As the regulator requires real time information from the other processes are different processes divided into different threads to allow for simulations calculation on one computer. There are five threads in total that handle different areas of the regulator. The system architecture for the regulator can be seen in Figure 5.1 and contains the following

1. **Regulator thread:** The regulator thread contains the main regulator algorithm that provides the real and virtual robots with their control signals in the form of joint torques $U1$ respective $U2$. The algorithm inside the regulator thread requires information from both robots in the form of joint angles $Q1$ and $Q2$, joint velocities $DQ1$ and $DQ2$ as well as the forces and torques that affect the robot's end-effectors $HE1$ and $HE2$.

2. **Visualization thread:** The visualization thread visualizes the virtual robot using a simple stick model depicting its links. This requires the joint angles for the virtual robot $Q2$.

3. **Communication thread:** The communication thread transfers the control signal from the **Regulator thread** to the **ur-rtde communication script** as well as reading the joint angle `actual_q` and joint angular velocity `actual_dq` from the real robot before providing them to the **Regulator thread**.

4. **Virtual robot simulation thread:** Inside the virtual robot simulation thread is the virtual robot arm is modelled. The joint torques $U2$ are supplied to the thread from the **Regulator thread** and drive the model that procures $Q2$, $DQ2$ and $HE2$ for the **Regulator thread** and **Visualization thread**.

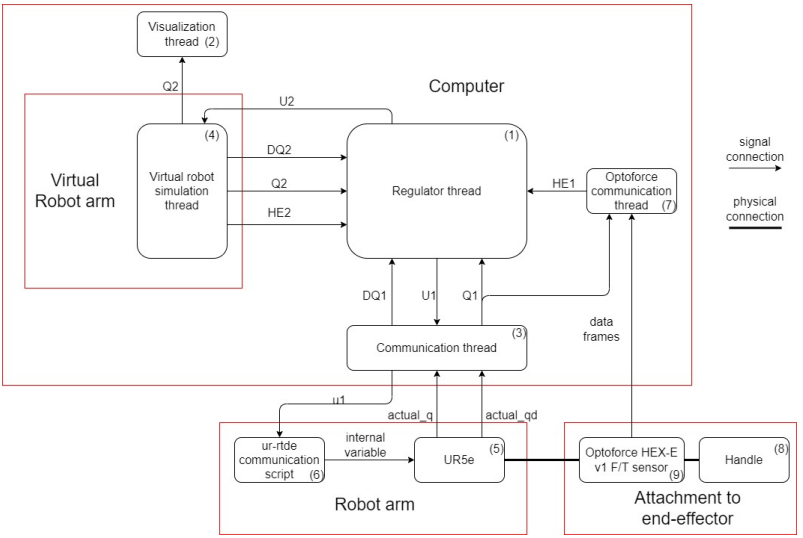


Figure 5.1 System architecture of the setup and what information is passed between the different parts of the setup. The setup consist of two parts, a computer that runs the virtual robot arm as well as the regulator and a UR5e robot arm where the Optoforce HEX-E v1 force-torque sensor is connected before the handle.

5. **UR5e robot:** The UR5e robot is the real robot arm of the algorithm providing the **Communication thread** with its joint angle `actual_q` and joint angular velocity `actual_dq` while having the value of its joint torque provided by the **Communication thread** $u1$. The **Optoforce HEX-E v1 F/T sensor** is mounted on the flange of the UR5e's end-effector.

6. **ur-rtde communication script:** The ur-rtde communication script is a script that is uploaded to the UR5e robot that instructs the UR5e robot how to interpret and use the torque $U1$ provided to it by the **Communication thread**.

7. **Optoforce communication thread:** The Optoforce HEX-E v1 F/T sensor produces data frames that need to be interpreted in the Optoforce communication thread before they can be used by the **Regulator thread**. The internal components of the F/T sensor are affected by gravity and these effects are cancelled out here which requires information about the rotation of the sensor in the form of joint angles from the **UR5e robot** to cancel out the gravity dependency.

8. **Handle:** The handle is a simple 3D printed construction allowing an operator to interact with the end effector of the **UR5e robot**.

9. **Optoforce HEX-E v1 F/T sensor:** The Optoforce HEX-E v1 F/T sensor is connected between the flange of the UR5e's end-effector and the handle. This allows it to measure the torques and forces that are applied to the end effector through the handle and transmit them to the **Optoforce communication thread** using data frames.

5.2 UR5e robot

The UR5e robot is a six dof robot with a max payload of 5 kg and a reach of 850 mm [*UR5e Information* 2020]. It is connected to the computer using an Ethernet cable with package rate of 500 Hz.

5.3 Virtual robot

The virtual robot was implemented according to the inverse dynamics and zero-order hold double integration of Chapter 4.4. The step size was set to $h = 0.001$ s and the mass parameters from the identification turned down to 25% of the real robot's values. The change of the mass parameters was implemented as a response to the low regulator frequency. This problem resulted in unintuitive forces and torques on the real robot as the regulator thread were unable to keep up with movements in shifting directions. In order to mitigate this problem was the mass of the virtual robot reduced, reducing the virtual forces on the real robot from the inertia of the virtual robot and therefore the unintuitive forces and torques on the real robot. This was done during the identification process of the real robot to decouple the measured parameters and then reduce the mass elements to 25% of their original value. This change does not completely correspond to a simple change in mass as many of these parameters are identified together. The small impact on the model was considered to be acceptable.

5.4 FT sensor

The UR5e robot does have an integrated F/T sensor, however, there are some practical problems with it. The result of these shortcomings was that an external Optoforce HEX-E v1 F/T sensor was installed on the flange of the UR5e's end-effector and the handle. The F/T sensor and the handle can be seen in Figure 5.2 and the mounting can be seen in Figure 5.3.

Figure 5.2 Optoforce HEX-E v1 F/T sensor and handle used.

UR5e integrated FT sensor

The raw sensor data from the integrated F/T sensor in the UR5e robot can not be accessed and then compensated for the given weight of the attached tool. The included compensation leaves a lot to be desired as it has trouble with shifting orientations. This results in inconsistent body frames and forces in directions that are not correct.

An external F/T sensor was chosen as the internal F/T sensor is not capable to perform its task.

Optoforce HEX-E v1 force torque sensor

The Optoforce HEX-E v1 F/T sensor is a 6-axis F/T sensor with a default sampling rate of 100 Hz. The nominal capacity and deformation of the F/T sensor can be found in Table 5.1. The force and torque values from the Optoforce F/T sensor are

	Nominal Capacity	Deformation
Force x, y-directions	$\pm$200 N	$\pm$1.7 mm
Force z-direction	$\pm$200 N	$\pm$0.3 mm
Torque x, y-directions	$\pm$10 Nm	$\pm 2.5^o$
Torque z-direction	$\pm$6.5 Nm	$\pm 5^o$

Table 5.1 Normal capacity and deformation for the Optoforce HEX-E v1 F/T sensor [optoforce, 2017]

given in the dimensionless counts and converted to N and Nm using the sensitivity

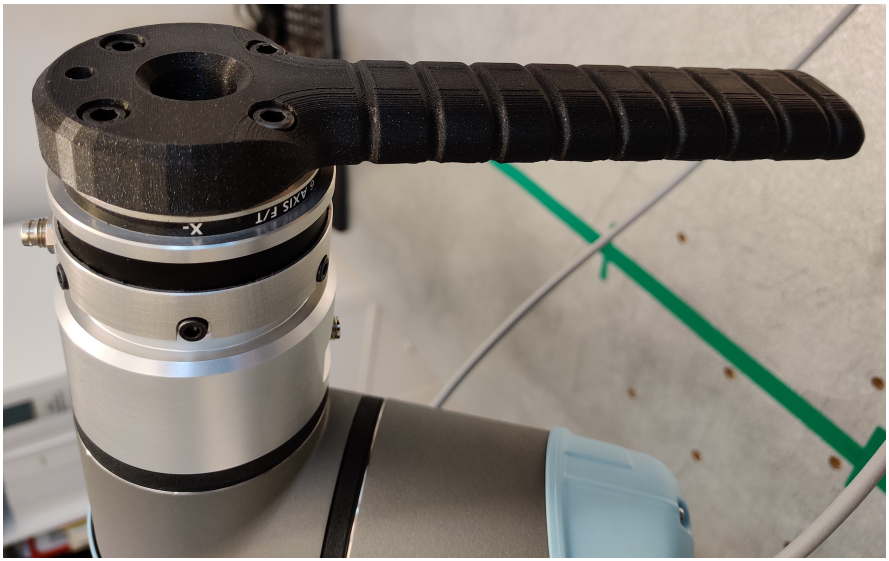


Figure 5.3 Mounting of the external Optoforce HEX-E v1 F/T sensor on the flange of the UR5e's end-effector with the handle installed on the F/T sensor

(counts at normal capacity, countsN.C) and the nominal capacity (N.C) using the expression in [*OptoForce General DAQ - protocol description 1.4 - USB, CAN, UART* 2015].

As an example for F_y

$$F_y[N] = F_y[Counts]/(countsN.C) \cdot (N.C) \tag{5.1}$$

The sensitivity and nominal capacity values used can be seen in Table 5.2.

	Nominal Capacity	Counts at Normal Capacity
F_x[N]	200	2000
F_y[N]	200	2000
F_z[N]	200	2000
T_x[Nm]	10	10000
T_y[Nm]	10	10000
T_z[Nm]	6.5	6500

Table 5.2 Normal capacity and counts at normal capacity used for transforming the dimensionless count from the Optoforce HEX-E v1 F/T into N and Nm. These values are based on the sensitivity report in Appendix D. To transform the torque into Nm, the nominal capacity was divided by 1000.

However, the values calculated by Equation (5.1) using the values in Table 5.2 were subject to three main problems.

- Inherent offsets and drift of these offsets for the different axes.
- Influence of gravity on the internal components of the F/T sensor.
- Influence of gravity on the handle attached to the F/T sensor.

The inherent offsets and drift of these offsets for the different axes were compensated by first isolating the influence of gravity on the internal components of the F/T sensor. With influence of gravity on the internal components of the F/T sensor compensated for the F/T sensor was placed with the force of gravity pointing in the negative z-direction in the F/T sensor's frame of reference [optoforce, 2017]. The remaining offsets were considered the inherent offsets for the different axis. However, the drift of the F/T sensor offsets results in any calibration only working well for roughly 4 hours. This was solved by using the first 100 values from the F/T sensor for approximating the drift and compensating for it. The only thing left was to compensate for the influence of gravity on the handle attached to the F/T sensor. As the influence of gravity on the internal components of the F/T sensor already were calculated the influence of the internal components in the F/T sensor and the influence on the handle attached to the F/T sensor were compensated for separately. These influences were compensated for by the rotation of the F/T sensor in relationship with the space frame using the adjoint representation in Equation (3.48).

5.5 Tuning the regulator

The tuning of the implemented algorithm is done by four matrices implemented into the state variable feedback in Equation (3.21). Each of the four constants K_{tp}, K_{op}, K_{td} and K_{od} are made of a 3×3 diagonal matrix where the diagonal elements is connected to one direction in the base frame of reference. This allows tailoring of the implemented algorithm based on the specific direction in the base frame of reference as well as to specify the response to transitional and orientation errors separately.

The state variable feedback results in a second-order system using ILM Modelling [*ILM for Basic Modeling and Identification* 2020] was visualized and can be seen in Figure 5.4.

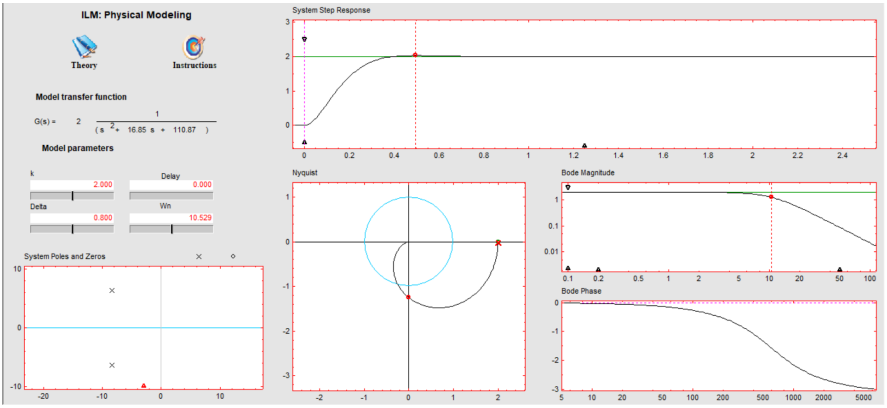


Figure 5.4 Visualization of the state variable feedback second order system using ILM for Basic Modelling and Identification. A settling time of 0.5 seconds was chosen, and the model transfer function was then used to derive the constants for the calibration of the implemented algorithm.

Testing different settling times resulted in a settling time of 0.5 seconds resulting in the best combination of small error and user-friendliness resulting in a proportional constant $k_t = 110.87$ and the derivative constant $k_p = 16.85$. These were implemented for all directions in the base frame of reference with the same response to transitional and orientation errors as

$$K_p = \begin{pmatrix} 110.87 & 0 & 0 & 0 & 0 & 0 \\ 0 & 110.87 & 0 & 0 & 0 & 0 \\ 0 & 0 & 110.87 & 0 & 0 & 0 \\ 0 & 0 & 0 & 110.87 & 0 & 0 \\ 0 & 0 & 0 & 0 & 110.87 & 0 \\ 0 & 0 & 0 & 0 & 0 & 110.87 \end{pmatrix}$$

$$K_d = \begin{pmatrix} 16.85 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16.85 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16.85 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16.85 & 0 & 0 \\ 0 & 0 & 0 & 0 & 16.85 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16.85 \end{pmatrix} \tag{5.2}$$

5.6 Summary

The experimental setup consists of a UR5e robot connected to a laptop with an Ethernet cable communicating at 500 Hz. On the flange of the UR5e robot, an Optoforce HEX-E v1 force-torque sensor was connected and on the flange of the F/T sensor the teaching handle was connected. There five threads active on the laptop that handle different parts of the regulator. These are

- A regulator thread that contains the algorithm.
- A communications thread that handles the communication between the UR5e robot and the regulator thread.
- An Optoforce communications thread that interprets the signals from the F/T sensor and transforms them into forces and torques which the regulator thread can use.
- A virtual robot simulation thread that simulates the virtual robot for the regulator thread.
- A visualization thread that visualizes the virtual robot with a simple stick model.

The information about forces and torques measured by the F/T sensor is interpreted by the Optoforce communications thread, the values are compensated for the inherent offset and drift of the sensor as well as the influence of gravity on it and the handle attached.

6

Test description

Five main tests were conducted to test the implementation of the algorithm. These were

- **Free space tracking:** During this test the end effector was placed in different positions and the tracking of the virtual robot investigated as well as the virtual forces of the regulator were investigated. One of these positions placed the virtual and real robot in a singular configuration to test its reaction to the singularity.
- **Virtual Forces:** During this test the end effector was placed in different positions and the virtual forces of the regulator were investigated in context with the movement of the end effectors.
- **Surface collision:** During this test a virtual wall was implemented and the haptic response of the real robot was investigated when the virtual robot collides with the virtual wall.
- **Physical limitation:** During this test the virtual robot was placed in a configuration close to its physical limitation and the real robot were then moved in an attempt to extend the virtual robot beyond its physical limitations.
- **Influence of F/T sensor:** During this test both robots were placed in the same start configuration and the end effector then moved to different positions. The same movements was then emulated without the F/T sensor contributing to the algorithm. The error, position and forces were then compiled to investigate the influence of the F/T sensor on the algorithm.

6.1 Free-space tracking

The free-space tracking test was based on nine movements. The first three were small rotations of the outer joints followed by movements in the x, y and z directions of the base frame. The start configuration can be seen in Figure 6.1. The movements were the following

- 90° rotation for joint 5
- 90° rotation for joint 4
- 90° rotation for joint 6 (no translation)
- Movement in the negative y-direction
- Movement in the positive y-direction
- Movement in the positive z-direction
- Movement in the negative z-direction
- Movement in the negative x-direction
- Movement in the positive x-direction

The movement in positive z-direction results in a singularity configuration as the arm is fully extended. This was chosen as a test for singularity-free motion. Here the TCP (Tool Center Point), the error and the forces and torques recorded by the F/T sensor investigated.

6.2 Virtual forces

The free-space tracking test was based on six movements in the x, y and z directions in the base frame. The start configuration can be seen in Figure 6.1. The movements were as follows

- Movement in the negative x-direction
- Movement in the positive x-direction
- Movement in the positive z-direction
- Movement in the negative z-direction
- Movement in the positive y-direction
- Movement in the negative y-direction

Here the TCP and the virtual forces were investigated.

6.3 Surface collision

The surface collision test was based around movement from the start configuration seen in Figure 6.2 into the virtual wall seen in cyan. The forces and torques on the F/T sensor, the error as well as the TCP of both robots did then show the haptic response from the wall. For reference the repulsive forces from the virtual wall on the end effector for the virtual robot will also be investigated. For clarity a shaded area was be included in the graphs where the virtual robot was inside the virtual wall.

6.4 Physical limitation

The physical limitation test was based on extending the end effector of the virtual robot from the start configuration that can be seen in Figure 6.3 into its maximum reach. Then the movement in positive y-direction continued, trying to extend the virtual robot arm beyond its physical limitation. This was investigated by the forces and torques on the F/T sensor, the error as well as the TCP of both robots. A picture was included as an example of the maximum extension. For clarity was a shaded area included in the graphs where the virtual robot was extended to its maximum length.

6.5 Influence of F/T sensor

The influence of F/T sensor test were based on six movements in the x, y and z directions of the base frame. The start configuration differs as the handle were rotated for maximum control of the end effector. The movements were the following

- Movement in the positive z-direction
- Movement in the negative z-direction
- Movement in the negative x-direction
- Movement in the positive x-direction
- Movement in the positive y-direction
- Movement in the negative y-direction

The start configuration of movements in the z-direction can be seen in Figure 6.4. The start configuration of movements in the x-direction can be seen in Figure 6.5. The start configuration of movements in the y-direction can be seen in Figure 6.6.

Here the TCP, the error, the forces and torques recorded by the F/T sensor were investigated.

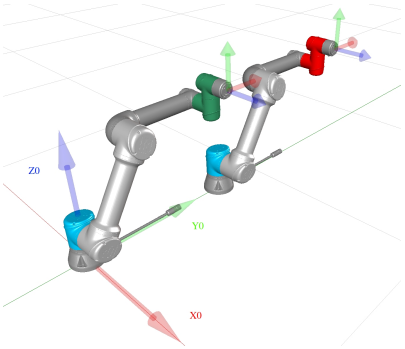


Figure 6.1 Start configuration used by free-space tracking and Virtual forces tests. The real robot is marked with a green coloured head while the virtual robot is marked with a red coloured head. The virtual robot is placed 0.75 m from the real robot in the y-direction. The base frame is marked as $\{0\}$ frame. The body frame has been included at the end effector of both robots as its orientation is hard to visualize.

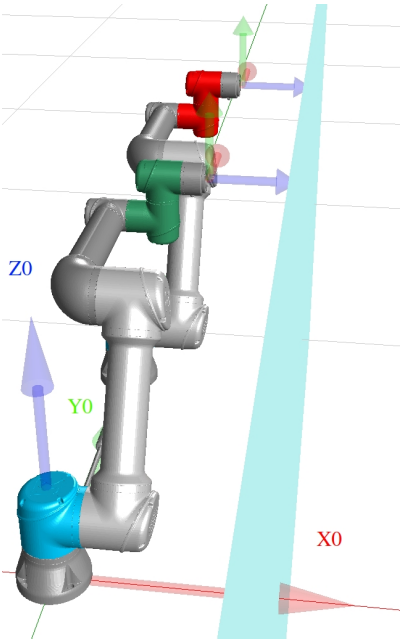


Figure 6.2 Start configuration used by the surface collision test. The real robot is marked with a green coloured head while the virtual robot is marked with a red coloured head. The virtual robot is placed 0.75 m from the real robot in the y-direction. The start configuration is the same as in the free space tracking test with the virtual wall included in cyan at $x = 0.45$. The base frame is marked as $\{0\}$ frame. The body frame has been included at the end effector of both robots as its orientation is hard to visualize.

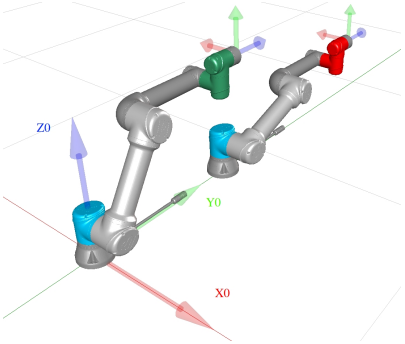


Figure 6.3 Start configuration used by the physical limitation test. The real robot is marked with a green coloured head while the virtual robot is marked with a red coloured head. The virtual robot is placed 0.75 m from the real robot in the y-direction. The base frame is marked as {0} frame. The body frame has been included at the end effector of both robots as its orientation is hard to visualize.

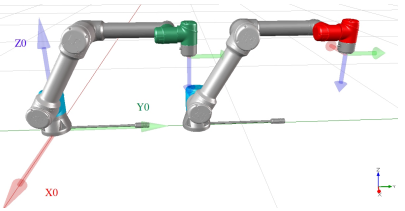


Figure 6.4 Start configuration used by the first z-direction movements in the influence of F/T test. The real robot is marked with a green coloured head while the virtual robot is marked with a red coloured head. The virtual robot is placed 0.75 m from the real robot in the y-direction. The base frame is marked as {0} frame. The body frame has been included at the end effector of both robots as its orientation is hard to visualize.

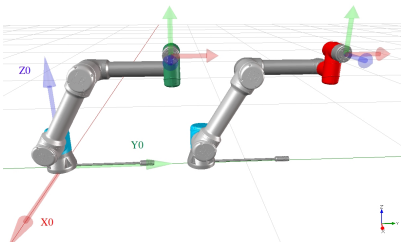


Figure 6.5 Start configuration used by the second x-direction movements in the influence of F/T test. The real robot is marked with a green coloured head while the virtual robot is marked with a red coloured head. The virtual robot is placed 0.75 m from the real robot in the y-direction. The base frame is marked as {0} frame. The body frame has been included at the end effector of both robots as its orientation is hard to visualize.

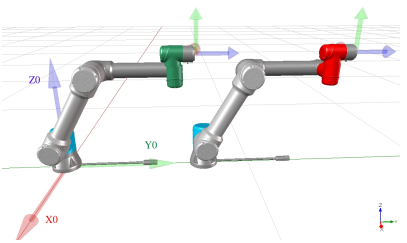


Figure 6.6 Start configuration used by the final y-direction movements in the influence of F/T test. The real robot is marked with a green coloured head while the virtual robot is marked with a red coloured head. The virtual robot is placed 0.75 m from the real robot in the y-direction. The base frame is marked as {0} frame. The body frame has been included at the end effector of both robots as its orientation is hard to visualize.

7

Result and discussion

7.1 Free space tracking

Figure 7.1 shows that the motion tracking in free space was archived with only a slight overshot for larger motions. The only thing of real note is the reaction of the virtual robot on the third motion at 30 seconds that is a rotation 90° around joint 6. This rotation does not result in a change of position for the real robot's TCP but results in a small change of position for the virtual robot's TCP.

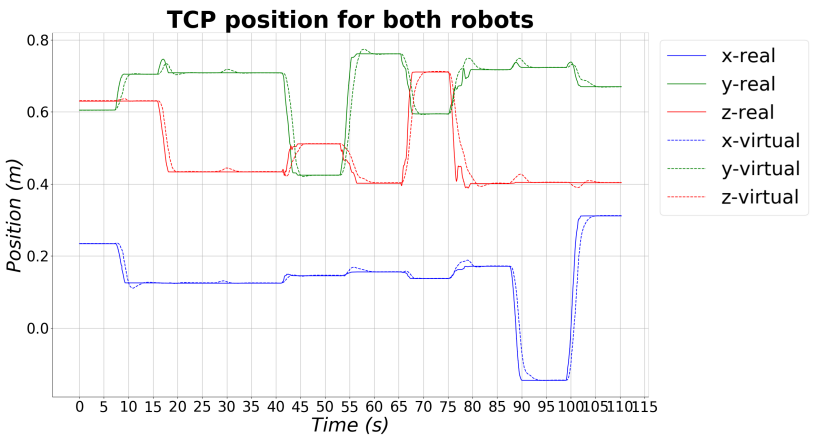


Figure 7.1 The TCP of both the real and the virtual robot during the free space tracking test. The x, y and z values are on the base coordinates seen from the origin in the base of the real robot. The distance between the real and virtual robots has been removed to better illustrate the tracking of the virtual robot arm. The singular configuration is reached at the time 70 seconds.

This is probably due to the model of both robots are not perfect. In theory, given a perfect model the rotational and translational parts are completely detached. However, the model identified for the real robot is not perfect. The virtual robot might not be perfect either as the implemented model for the virtual robot is based on an identification of a real robot and then modified so that the mass elements were reduced to 25% of real robots values. The impact of reducing the mass of the virtual robot might be larger than expected. The combined errors in both models might result in the rotational and translational having a small connection, resulting the small change of position for the virtual robot's TCP.

Another possible reason for this unexpected movement could be inaccuracies in the calibration of the F/T sensor. This might result in joint torques large enough to slightly move the lighter virtual robot but not large enough that the haptic response were felt or had any effect on the heavier real robot.

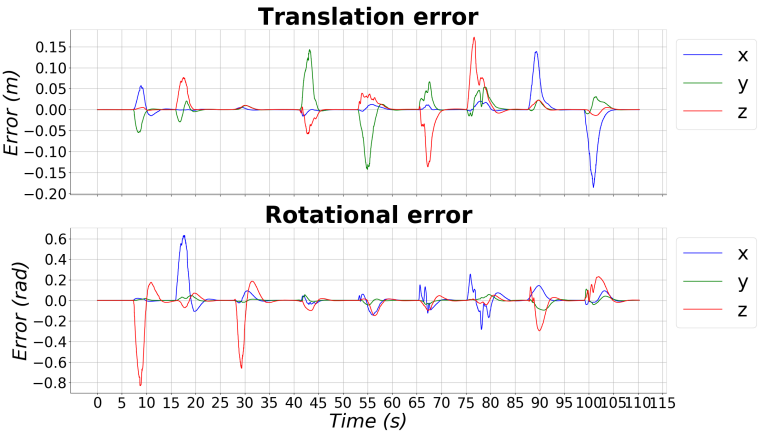


Figure 7.2 The error between the end effector of the real and the virtual robot during the free space tracking test. The rotational error has been transformed into XYZ intrinsic Euler angles for ease of understanding. The singular configuration is reached at the time 70 seconds.

The error between the real and virtual robot end effector can be seen in Figure 7.2. Here the rotational error can be seen for the first three movements and put in context with the residual rotational errors that happen when simple translational movements are attempted.

One way of reducing these errors is to increase the aggressiveness of the implemented algorithm by increasing the tuning constants in Equation (5.2). This however has to be weighed against the less intuitive and user-friendly experience that results in. The configuration of the real robot depends more on the configuration of

the virtual robot just as the other way around when the implemented algorithm is made more aggressive. The result is that the real robot actively fights back during standard motions in free space when one attempts to move it in a direction that the virtual robot is not already travelling in. This is in turn mainly due to the delay between the real and virtual robots, resulting in times where the virtual robot dictates the movements of the real robot.

So if the low frequency of the regulator thread can be solved and the delay between the real and virtual robot can be lessened then a more aggressive regulator can be implemented without the need to trade off with the user experience. More on this problem can be found in the Chapter 7.6.

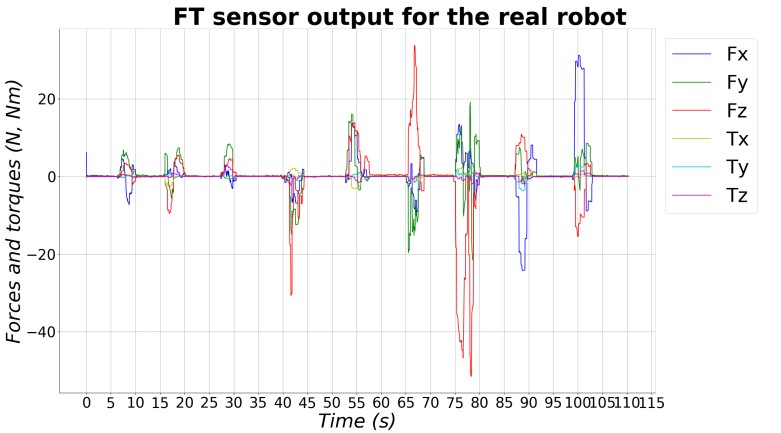


Figure 7.3 The F/T sensor output the end effector of the real robot expressed in the base frame situated in the base of the real robot during the free space tracking test. The singular configuration is reached at the time 70 seconds.

In Figure 7.3 the inputs are from the F/T sensor shown. As the force necessary for moving the end effector of the real robot depends on the configuration so the needed force to move the end effector in different directions are of different amplitude. The one standing out as a prime example of this is the increased force in the negative z-direction need to move the robot in negative y-direction at 42 seconds. Other than that the movements in the y-direction, a clear connection between the direction of the force and the movement of the TCP can be established.

At 70 seconds in the test were both robots were placed in a singular configuration to test singularity-free operation, this configuration can be seen in Figure 7.4. The configuration was reached and moved away from without problems.

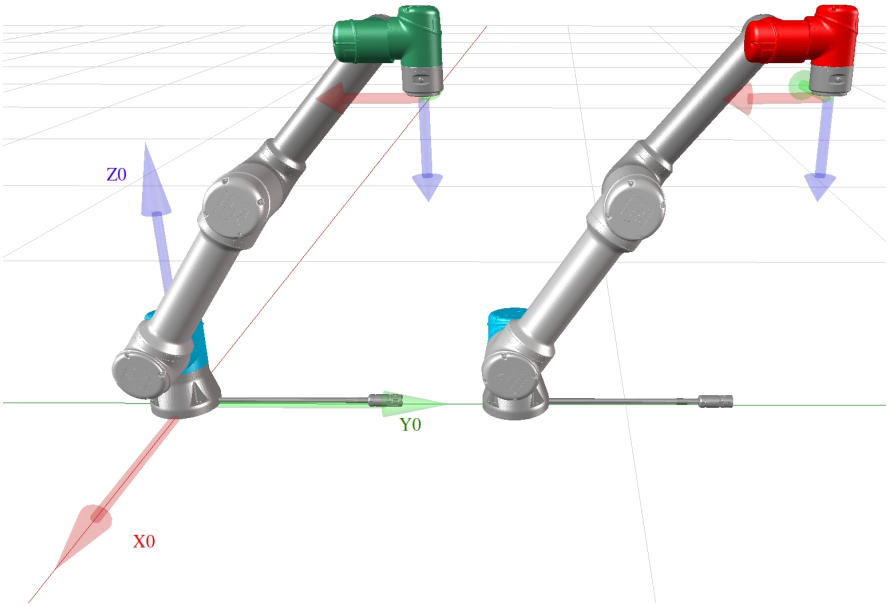


Figure 7.4 Picture taken from the Maplesim visualization of both the real and virtual robot during the free space tracking test. The real robot is marked with a green coloured head while the virtual robot is marked with a red coloured head. The virtual robot is placed 0.75 m from the real robot in the y-direction. The base frame is marked as $\{0\}$ frame. The body frame has been included at the end effector of both robots as its orientation is hard to visualize. This picture is taken at 72 seconds and depicts the configuration after both robots moved in the positive z-direction. This configuration was chosen as it is a singularity.

Based on the error in Figure 7.2 and the motions of the TCP for both robots in Figure 7.1 singularity free motion tracking in free-space has been archived.

7.2 Virtual forces

The virtual forces applied to the end effectors of the real and virtual robots can be seen in Figure 7.5 and Figure 7.6 respectively. The virtual forces are placed above the TCP of its motion for ease in seeing the connection between the two. As the real robot end-effector moves in the positive z-direction at 24 seconds in Figure 7.5 there is a virtual force pulling the real robot's end-effector back towards the virtual arm's end-effector. Reversely as the real robot's end-effector moves in the positive z-direction at 24 seconds in Figure 7.6 there is a virtual force pushing the virtual robot arms end-effector towards the real arms end-effector.

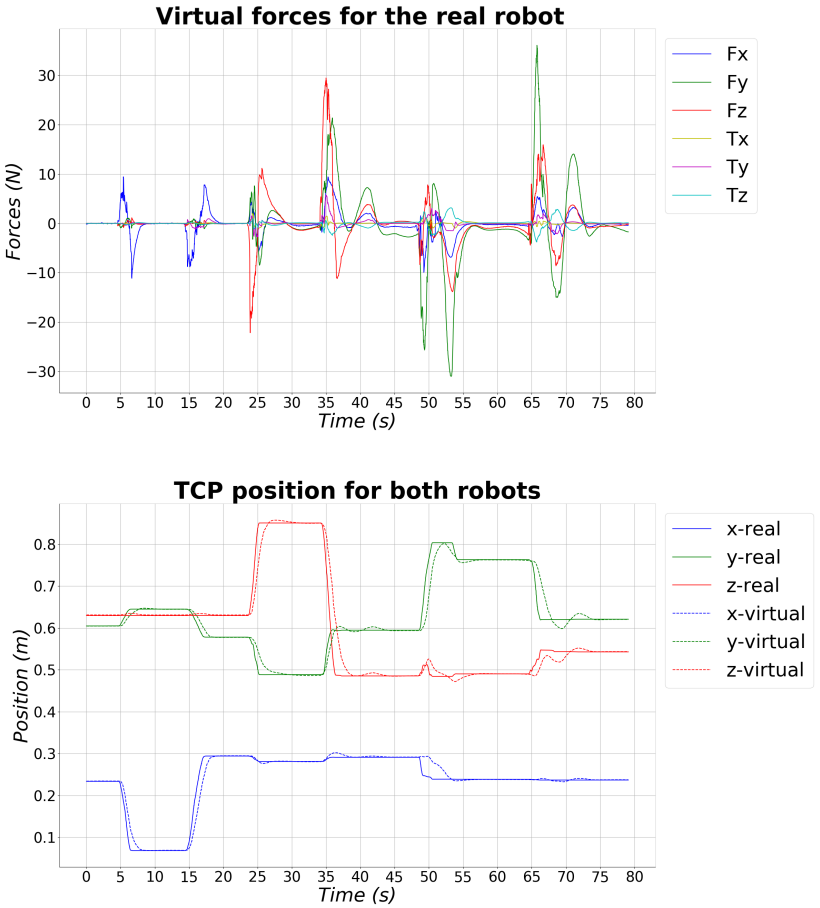


Figure 7.5 Virtual forces on the real robot expressed in the base frame situated in the base of the real robot. These are placed together with the TCP of both the real and the virtual robot during free space tracking tests for greater context of how the virtual forces result in the motion of the virtual robot. The x, y and z values are on the base coordinates seen from the origin in the base of the real robot. The distance between the real and virtual robots has been removed to better illustrate the tracking of the virtual robot arm.

When the real arm at 25 seconds in Figure 7.6 starts to slow down as it approaches its position, the virtual force affecting the virtual robot arm's end-effector reverses, forcing it to slow down and to approach the same position as the real robot's end effector. The opposite happens for the virtual forces of the real robot

arms at 25 seconds in Figure 7.5.

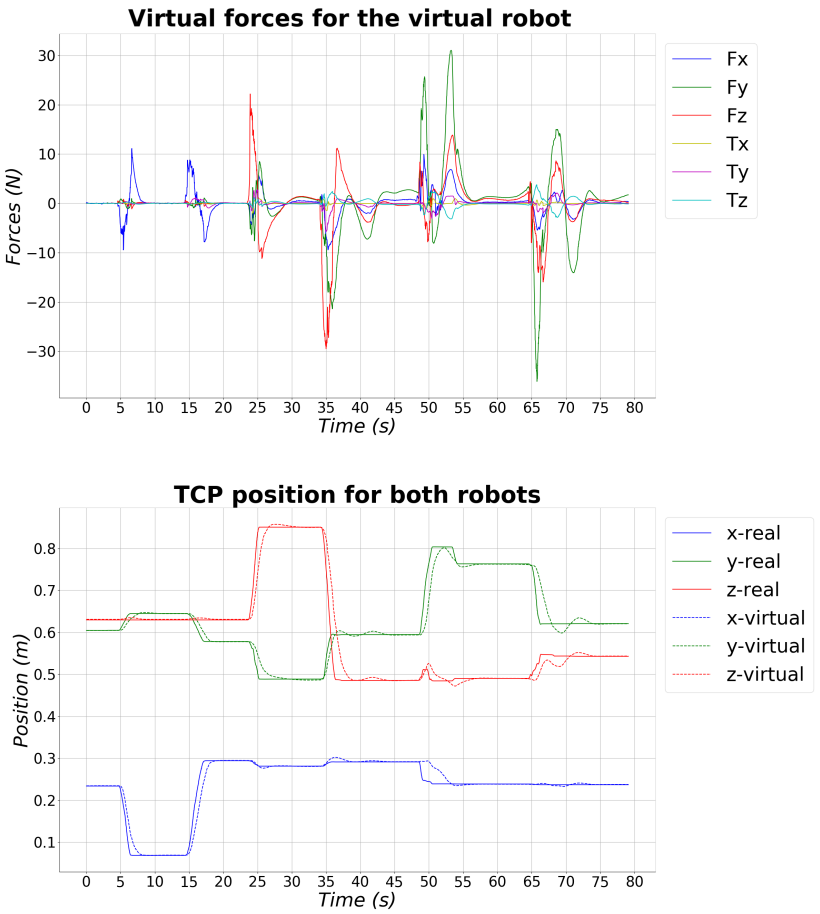


Figure 7.6 Virtual forces on the virtual robot expressed in the base frame situated in the base of the real robot. These are placed together with the TCP of both the real and the virtual robot during free space tracking tests for greater context of how the virtual forces result in the motion of the real robot. The x, y and z values are on the base coordinates seen from the origin in the base of the real robot. The distance between the real and virtual robots has been removed to better illustrate the tracking of the virtual robot arm.

Based on the correct directions of virtual forces in Figures 7.6 and Figures 7.5, have the correct response been achieved.

7.3 Surface collision

As seen in Figure 7.7 the TCP of the virtual robot is almost immediately stopped in the x-direction from reaching further into the virtual wall but with extensive force exerted on the real robot's end-effector it is still possible to reach deeper into the wall. There is movement in the y-direction as there are no forces counteracting movements in any other direction than x-direction as seen in Figure 7.8. This is expected as the wall is not solid but approximated with a spring force resulting in continuous motion into the virtual wall with an increasing repulsive force until the point where the handle risked breaking.

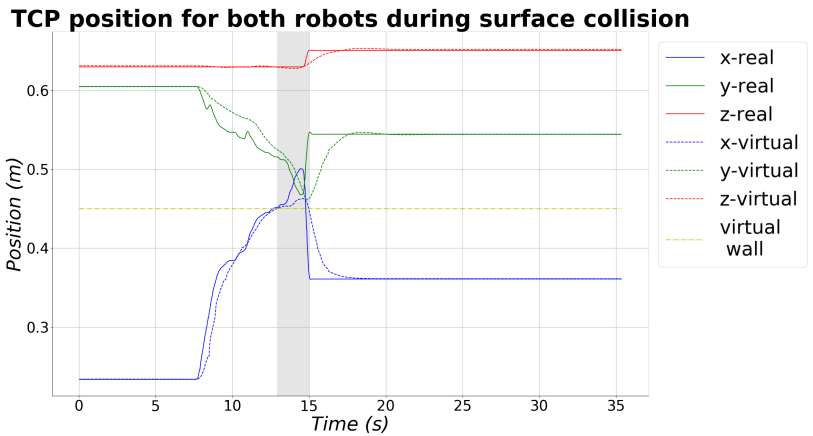


Figure 7.7 The TCP of both the real and the virtual robot during the surface collision test. The x, y and z values are on the base coordinates seen from the origin in the base of the real robot. The distance between the real and virtual robots has been removed to better illustrate the tracking of the virtual robot arm. The virtual surface that the virtual arm collides with is marked as the virtual wall in yellow at $x = 0.45$. The grey area indicates when the virtual robot is inside the virtual wall.

The distance that the virtual robot can enter into the virtual wall can be reduced by increasing the spring constant. During this experiment the spring constant was set to $k = 6000\ N/m$ but it can be increased further. The distance that the real robot can enter into the virtual wall is dependent on several factors, the aggressiveness of the implemented algorithm, the maximum deformation of the F/T sensor and the max counterforce that the UR5e robot is capable of producing. An increase in the aggressiveness of the implemented algorithm will reduce the error seen in Figure 7.9 but as stated in the discussions about free-space tracking with result in a less intuitive and user-friendly experience. There is also a maximum deformation of the

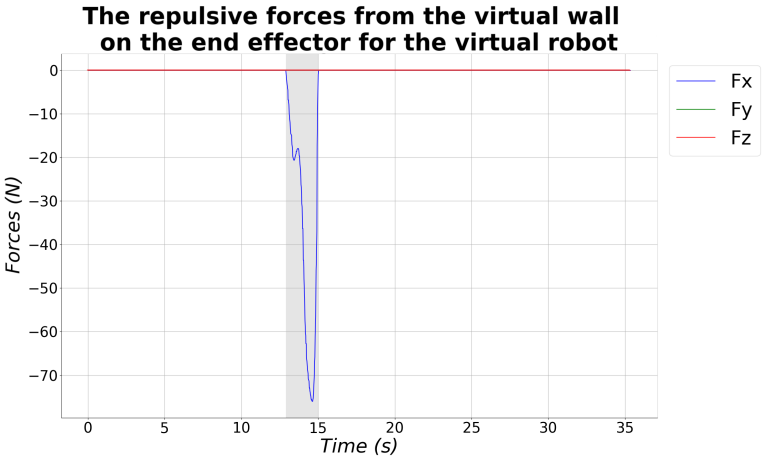


Figure 7.8 The repulsive forces on the end effector of the virtual robot are expressed in the base frame situated in the base of the real robot during the surface collision test. The grey area indicates when the virtual robot is inside the virtual wall.

F/T sensor to take into account when forcing the end effector of the real robot further into the virtual wall. A way of detecting when maximum deformation based upon saturation of the output signal were not implemented. Due to this were the no indicators on when this maximum deformation of the F/T sensor is reached, making it hard to theorize how far one could go before this becomes a limiting factor. This is a question tied to the max counterforce that the UR5e robot is capable of producing. It was proven during the testing of the physical limitations that it is possible to pulling with a greater force than the UR5e robot was capable of countering. This means that it is possible to extend the real robot further into the virtual wall than a stronger robot would allow. However, under the assumption that one does not try to force the real robot further into the virtual wall than the UR5e is capable of counteract in its current configuration, the main determining factors are the aggressiveness of the implemented algorithm, the maximum deformation of the F/T sensor and the durability of the handle. During several of the tests there were indications that the 3D printed handle might not be able to handle the tug of war between the counter forces from the algorithm and the external forces applied to the handle.

The error between the real and virtual robot end-effector during the time when the virtual arm was inside the virtual wall can be best seen in Figure 7.9 with a dominating translational error in the x-direction with an expected smaller component of translational error in the y-direction. These errors were then reversed as the handle was released and the counter forces ejected both the real and virtual robot from the virtual wall. This is due to the spring force that simulated an elastic sur-

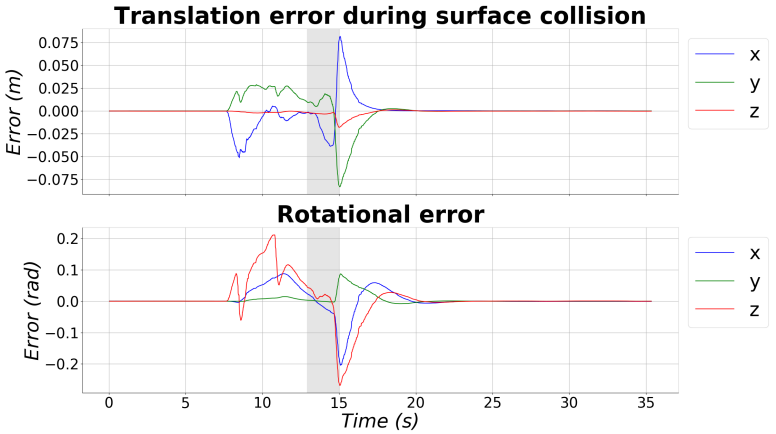


Figure 7.9 The error between the end effector of the real and the virtual robot during the surface collision test. The rotational error has been transformed into XYZ intrinsic Euler angles for ease of understanding. The grey area indicates when the virtual robot is inside the virtual wall.

face, for a solid surface there will be no ejection. One can also see an error in the z-direction in Figure 7.9 and the resulting movement in Figure 7.7. These are due to the joint configuration of the real robot that when given a force in mostly negative x-direction in the base frame as seen in Figure 7.10 gives way to a slight rotation around joints 4 and 5 while the majority of the rotation is confined to the shoulder joint 1. The graph over the forces seen in Figure 7.10 shows that a general movement in the x-direction of the base frame takes roughly 26 N of force in the robot's current configuration as the end effector of the real robot is set in motion. When the haptic forces from the algorithm started applying, the force required for continues movement into the virtual wall increased sharply until at 75.5 N I could not move the handle further without risking the durability of the handle.

Based on the forces in Figure 7.10 showing the haptic response from the implemented algorithm and the TCP of both robots in Figure 7.7 intuitive haptic responses have been achieved.

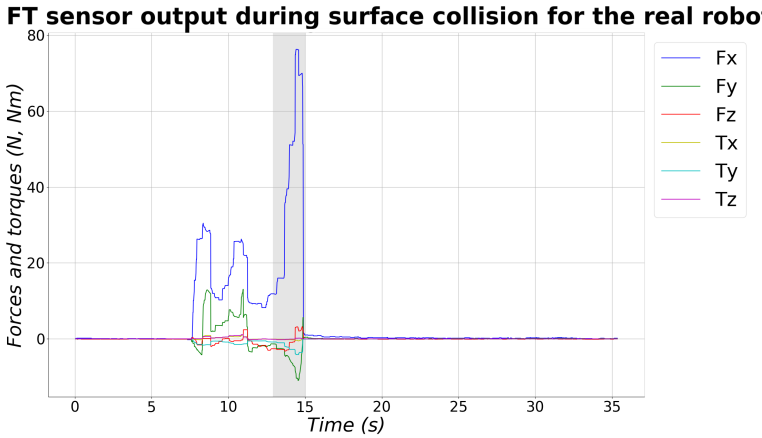


Figure 7.10 The F/T sensor output the end effector of the real robot expressed in the base frame situated in the base of the real robot during the surface collision test. The grey area indicates when the virtual robot is inside the virtual wall.

7.4 Physical limitation

In Figure 7.11 the TCP of the real and virtual robot are compared when the virtual robot arm reaching its physical limitation. The virtual robot was set out to reach its maximum parallel to the y-axis of the base frame, placing its end effector at 0.917 m in the y-direction. As the virtual robot reached its physical limitation and the real robot continued the algorithm attempted to twist the TCP so that to reach further in the y-direction, however this twist resulted in a rotation error as well as a translational error in the z-direction as seen in Figure 7.12. This forced the virtual arm back from its physical limitation, resulting in the retraction from the physical limitation seen between the three grey marked area. The virtual arm still moves slightly even when fully extended in response to the movements of the real arm as one attempts to pass beyond the restrictions from the virtual arm's physical limitations. These small movements of joints 1 and 2 are responsible for the internal deviations inside the grey marked areas. In the meantime the real arm resisted the attempts to move it further in the y-direction. During the attempts to move the real arm beyond the physical limitations of the virtual robot the real arm slid slightly in negative z-direction resulting in the slight change in y and z-position for the real robot's TCP.

The difference between the real and virtual robot's TCP as seen in Figure 7.12 can be made smaller with a more aggressive regulator. In this configuration it is possible to pull with a greater force than the UR5e is capable of producing, resulting in the possibility of pulling it past the point where a stronger robot would not

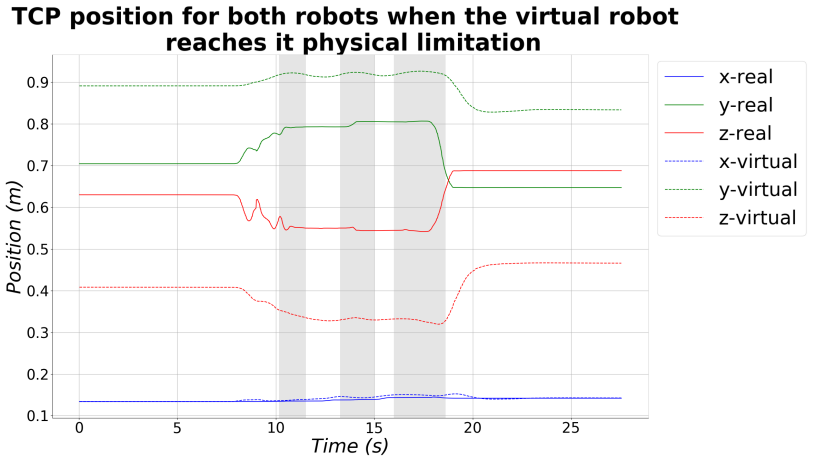


Figure 7.11 The TCP of both the real and the virtual robot during the physical limitation test. The x, y and z values are on the base coordinates seen from the origin in the base of the real robot. The distance between the real and virtual robots has been removed to remain consistent for all tests. The grey area indicates when the virtual robot is extended to its maximum length.

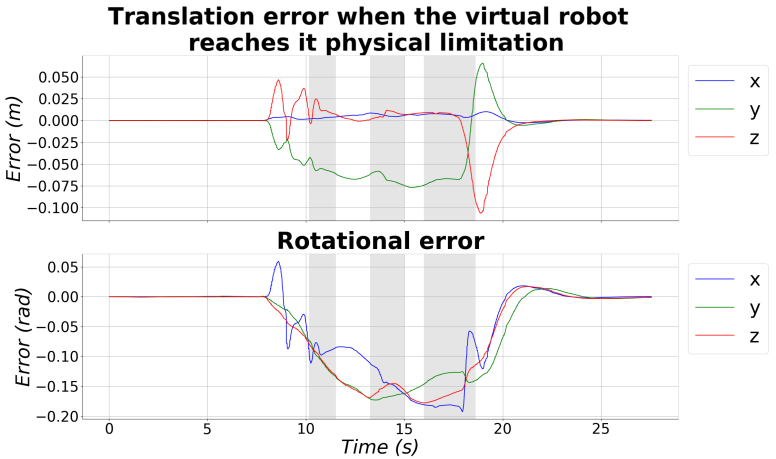


Figure 7.12 The error between the end effector of the real and the virtual robot during the physical limitation test. The rotational error has been transformed into XYZ intrinsic Euler angles for ease of understanding. The grey area indicates when the virtual robot is extended to its maximum length.

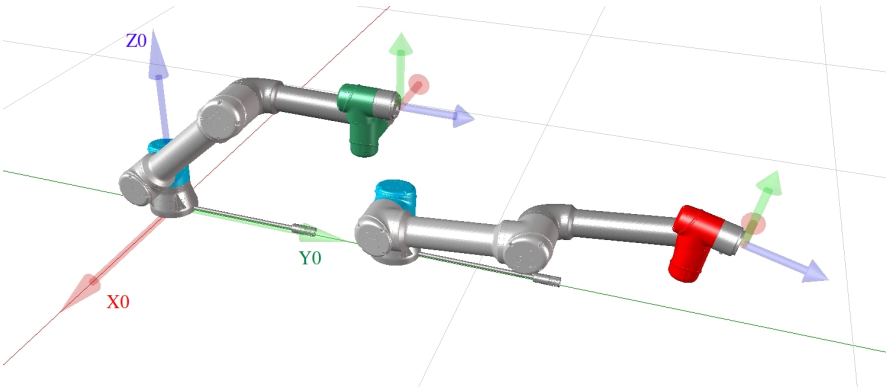


Figure 7.13 Picture taken from the Maplesim visualization of both the real and virtual robot during the physical limitation test. The real robot is marked with a green coloured head while the virtual robot is marked with a red coloured head. The virtual robot is placed 0.75 m from the real robot in the y-direction. The base frame is marked as $\{0\}$ frame. The body frame has been included at the end effector of both robots as its orientation is hard to visualize. This picture is taken at 17.272 seconds placing it approximately in the middle of the last grey belt in Figure 7.11 - 7.14.

have allowed it. The forces applied on the end effector of the real robot can be seen in Figure 7.14 where the attempts to pull the real robot beyond the physical limitations can be seen. As with the surface collision test the durability of the handle was a concern since the maximum repulsive force was generated by the UR5e. There are forces apparent in the x and z- directions of the base frame as the virtual arm attempted to remain relatively parallel to the y-axis in the base frame.

Based on the forces in Figure 7.14 showing the counterforce from the algorithm resulted from the physical limitation of the virtual robot and the TCP position seen in Figure 7.11 have the interaction forces when the virtual arm reached its physical limitations have been proven.

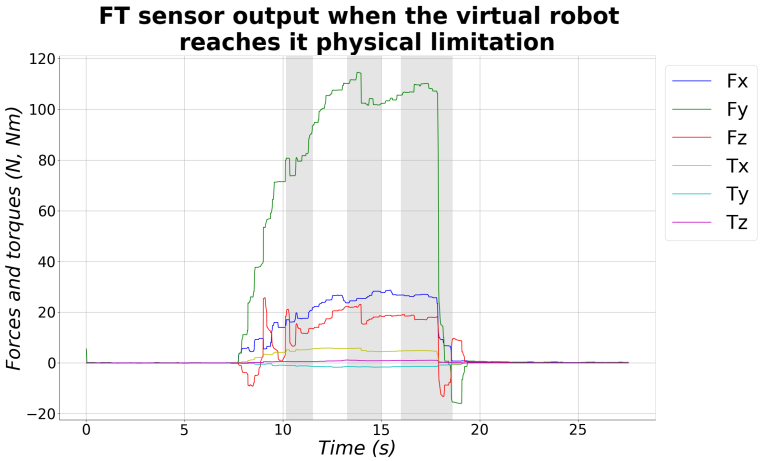


Figure 7.14 The F/T sensor output the end effector of the real robot expressed in the base frame situated in the base of the real robot during the physical limitation test. The grey area indicates when the virtual robot is extended to its maximum length.

7.5 Influence of F/T sensor

The TCP positions of both the real and virtual robots, with or without the implemented algorithm using the F/T sensor can be seen in Figure 7.15. As it is made up of six different subtests in an attempt to reproduce the same forces there are five discontinuations. Due to these tests not taking the same time the start of motions were different, but the end position after the motion was generally reproduced between the tests with and without the implemented algorithm using the F/T sensor. The two most clear differences in end-effector position are that the first movement was a movement in the positive z-direction at 10 seconds where the test without the implemented algorithm using the F/T sensor did not reach the target position and the fifth movement was a movement in positive y-direction at 50 seconds where the test without the implemented algorithm using the F/T sensor did not reach the target position. These errors show the fundamental flaw in this comparison. Although the position of the TCP can with reasonable accuracy be reproduced it is much harder to reproduce the same force and torque on the end effector. This makes comparing the response of the implemented algorithm with or without the implemented algorithm using the F/T problematic as the inputs between the two tests are not identical. This fundamental flaw is the most probable reason why the error in Figure 7.16 and output from the F/T sensor in Figure 7.17 is so similar with no clear conclusion to be drawn.

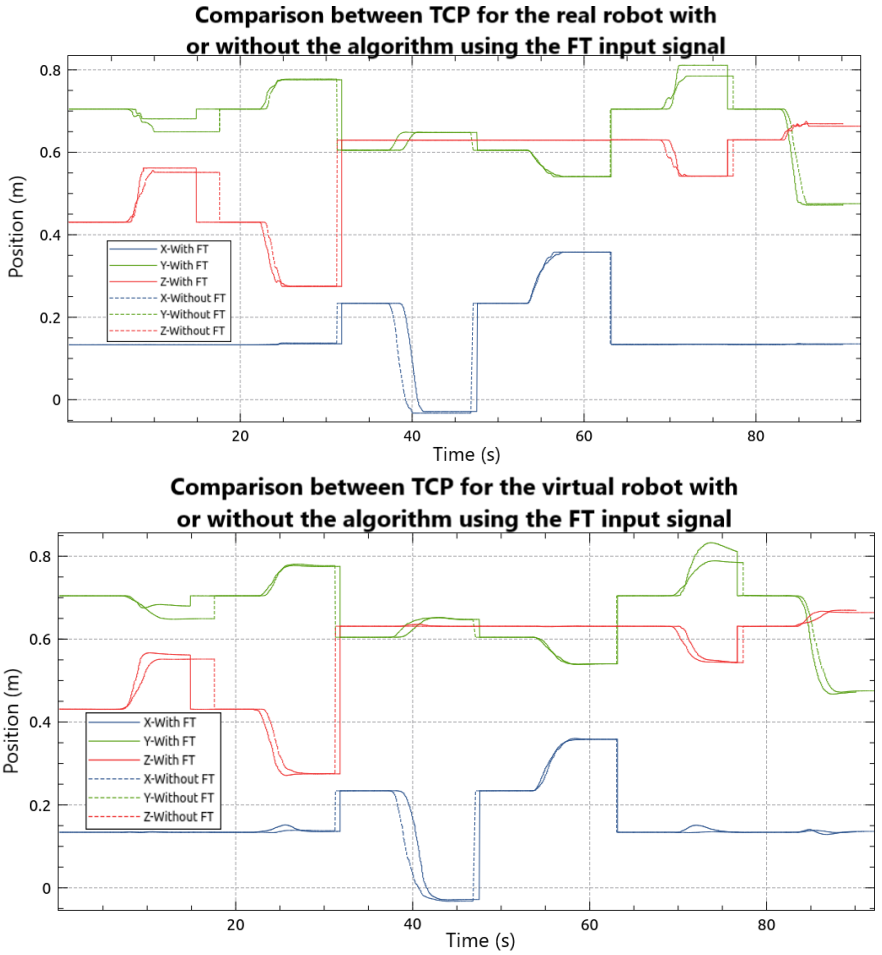


Figure 7.15 The TCP of both the real and the virtual robot during the influence of F/T sensor test. The x, y and z values are on the base coordinates seen from the origin in the base of the real robot. The distance between the real and virtual robots has been removed to better illustrate the tracking of the virtual robot arm. As the test was subdivided into single motions in an attempt to create a similar force with and without the influence of the F/T sensor are their jumps were one recorded motion transition into the next.

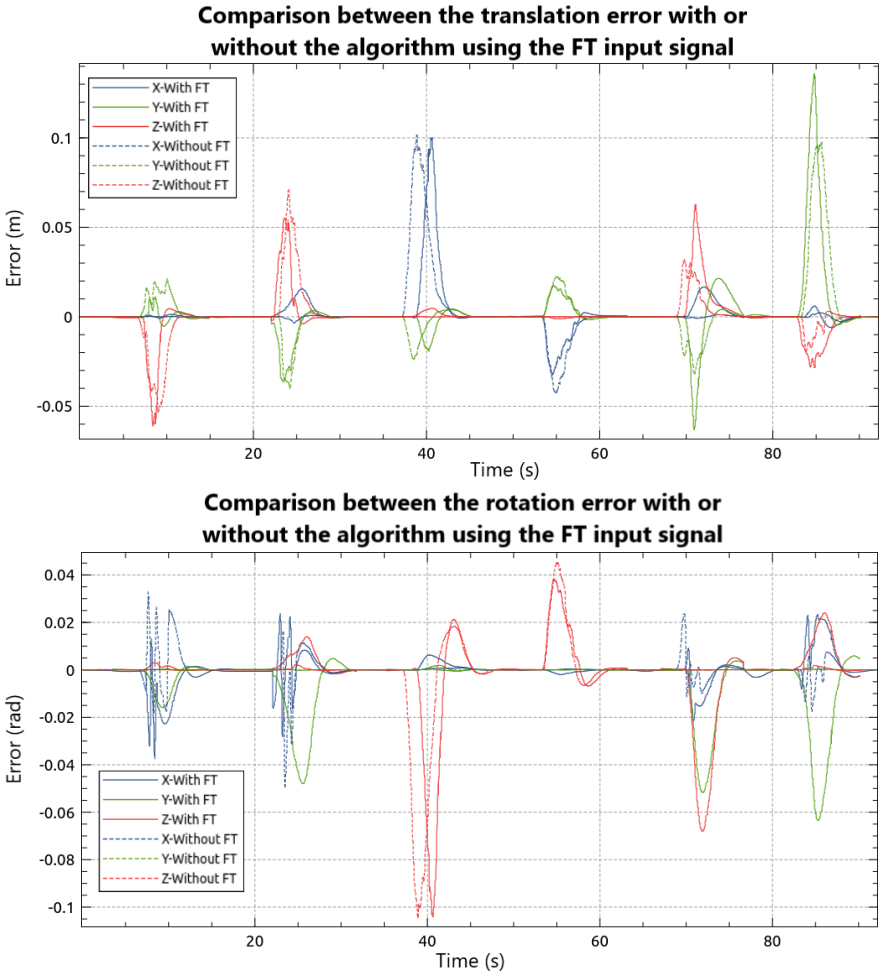


Figure 7.16 The error between the end effector of the real and the virtual robot during the influence of F/T sensor test. The rotational error has been transformed into XYZ intrinsic Euler angles for ease of understanding. The difference in synchronization between the different graphs is due to a small difference in the motions between the run with and without the F/T sensor.

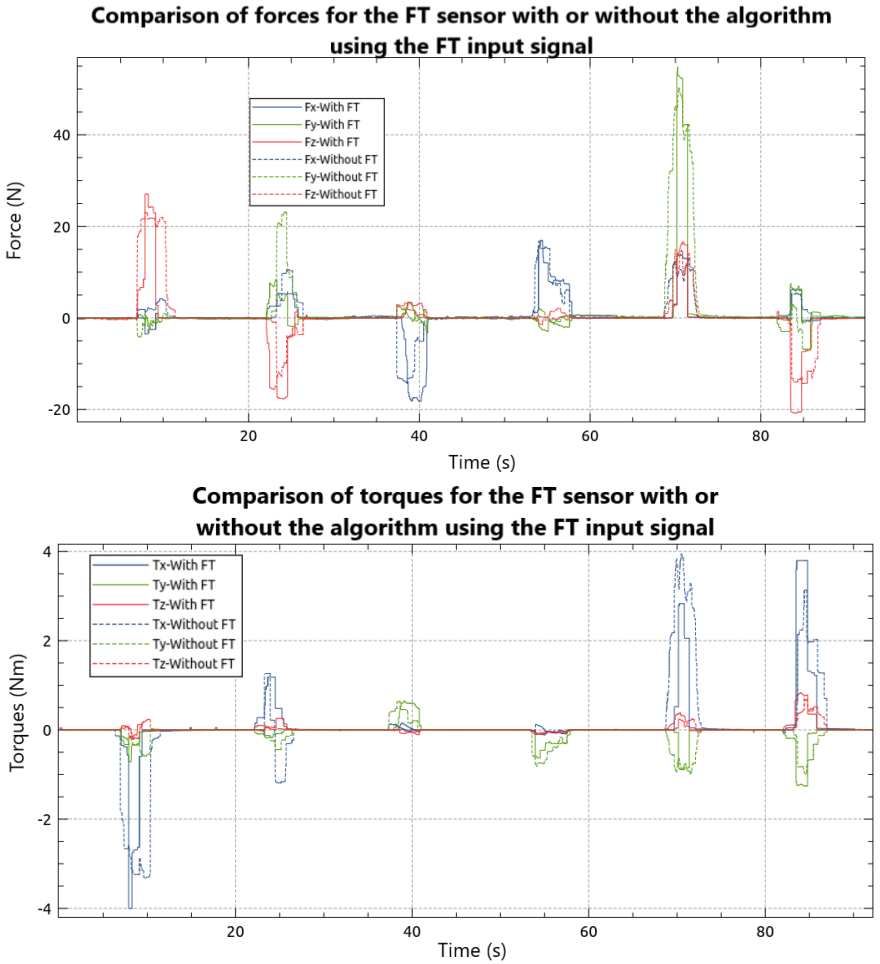


Figure 7.17 The F/T sensor output on the end effector of the real robot is expressed in the base frame situated in the base of the real robot during the influence of F/T sensor test.

Other possibilities are that the low mean frequency of 46.82 Hz for the regulator thread with the algorithm using the F/T sensor and 48.10 Hz for the regulator thread without the F/T sensor combined with the delays of the setup were severe enough to overshadow the influence of the F/T sensor. The difference between the mean frequency of the six movements with and without the implemented algorithm using the F/T sensor was within the range of difference that happens between different iterations, as seen in Table 7.1 and 7.2, this does not suggest that there is a slight improvement in the threads with the implemented algorithm using the F/T sensor. The parts that take computation time as the matrix calculation of the body Jacobian and its multiplication with the output of the F/T sensors were done in both cases.

Thread	FST	VF	SC	PL	With F/T	Without F/T
Regulator	47.14	46.93	46.41	47.73	46.82	48.10
Communication	119.19	119.31	121.09	118.24	119.31	120.21
Optoforce	86.02	87.63	87.42	87.38	85.63	84.96
Simulation	157.37	156.00	159.09	158.43	158.67	156.76
Visualization	1.20	1.31	1.01	0.92	0.96	1.07

Table 7.1 The mean frequencies of the different threads during the different tests in hertz (Hz). These tests were repeated 5 times, the variance of the five iterations can be found in Table 7.2. Note that the mean frequency for the influence of F/T sensor tests is based on the mean values of the six different movements inside each iteration. The abbreviations are as follows. Free space tracking test (FST), Virtual forces (VF), Surface collision test (SC), Physical limitation test (PL), Influence of F/T sensor with the implemented algorithm using the F/T sensor (With F/T), Influence of F/T sensor without the implemented algorithm using the F/T sensor (Without F/T)

Thread	FST	VF	SC	PL	With F/T	Without F/T
Regulator	0.91	0.83	0.92	0.90	0.76	0.81
Communication	0.62	0.62	0.71	0.81	0.90	0.92
Optoforce	1.02	0.67	1.24	0.84	0.95	1.07
Simulation	1.30	0.84	0.93	1.02	0.80	1.06
Visualization	0.10	0.11	0.10	0.09	0.12	0.13

Table 7.2 The variance of the different threads during the different tests in hertz (Hz). These tests were repeated 5 times, the mean of the five iterations can be found in Table 7.1. Note that the variance for the influence of F/T sensor tests is the mean values of the six different movements. The abbreviations are as follows. Free space tracking test (FST), Virtual forces (VF), Surface collision test (SC), Physical limitation test (PL), Influence of F/T sensor with the implemented algorithm using the F/T sensor (With F/T), Influence of F/T sensor without the implemented algorithm using the F/T sensor (Without F/T)

The main idea to mitigate the fundamental problem is to reverse the master, slave relationship of the real and virtual robots. Currently the real robot is the master robot and the virtual robot is the slave robot. However if that is reversed, the force inputs on the end effector of the virtual robot will be the catalyst for the movements. Using a separate regulator it would be possible to apply the same force to the end effector at the same time, creating an identical input signal with the only deviating factor being if the implemented algorithm is using the F/T sensor or not.

Based on the discussed flaws of the test and the information given by Figures 7.15, 7.16 and 7.17 it is not possible to demonstrate an improvement in the haptic experience with or without the implemented algorithm using the F/T sensor.

7.6 Error sources

Frequency

The first and most likely error source that affects all the above tests is the low frequency of the regulator thread. A table with the frequency for the different tests can be found in Table 7.1. This is most likely a code optimization problem with the largest known contributor being code created by Cognibotics. This code is generalized to work for a number of different robots and not specifically tailored for the UR5e robot. As an example, during the optimization done on the regulator thread, the homogeneous transformation matrix and the transformation from geometric to body Jacobian were implemented in the mathematical framework behind Maplesim, Maple [*Maple* 2020] and then transferred into Python code as a replacement for the code created by Cognibotics. These implementations did result in a reduction of the individual calculation time by a factor larger than ten. Continued tailoring of other code created by Cognibotics for instance, the creation of the geometric Jacobean, mass, Coriolis and centripetal matrices would improve the frequency even further but were not done in this work due to time constraints.

The communication thread uses Real-Time Data Exchange (RTDE) to communicate with the UR5e robot. RTDE generally generates output messages at 125 Hz. However, the real-time loop in the regulator has a higher priority than the RTDE interface. Therefore, if the regulator lacks computational resources it will skip a number of output packages [*RTDE* 2020]. This gives the communication thread a maximum frequency of 125 Hz for new information. There is not a clear path of optimization inside the communication thread as the runtime work is to transfer information from the UR5e into a format that the rest of the implementation can use. It might even be so that increasing the frequency might not improve performance as the ur-rtde communication script uses a UR-provided unofficial way of directly controlling the torques. As this is not one of the standard commands used by RTDE it is possible that the real-time loop inside the UR5e regulator needs more computational resources, resulting in a lower output frequency of messages. As the communication thread already is operating close to the maximum frequency of RTDE it is possible

that a faster communication tread might not improve the data frequency from the UR5e.

The Optoforce communication thread has the possibility for increased optimization of the calculation time connected to the transformation of ticks into forces and torques. However, as the runtime work is relatively computationally light was the time placed on optimization focused on other threads.

The simulation thread contains many of the same functions created by Cognibotics as were used in the regulator thread, these include the calculation of the mass matrix, geometric Jacobian, Coriolis and centripetal matrix as well as the inverse dynamics calculation. As these are only done for one robot and without the main regulator are threads frequency higher than the other and though it can be improved by the continued tailoring of the code for specific UR5e robots it is not seen as the bottleneck for the overall delay of the implemented algorithm.

The low frequency of the visualization thread is due to its rudimentary nature and thou lacking does not affect the frequency of other threads.

Tests with removing different threads do not show a substantial increase in the frequency of the remaining threads making the exclusion of one or the absorption of its function into other threads not worth it. The last thing to address regarding the frequency of the implemented algorithm is the processors of the laptop used, the use of a computer with faster processors or more cores might improve the frequency for all threads.

The influence of the F/T sensor

The implementation of the F/T would in theory provide the virtual robot with a indication the motion of the real robot resulting in the algorithm not simply reacting on the error but the ability to be proactive. This theoretical improvement was not observed under the influence of the F/T sensor test, and it is most likely due to the fundamental flaw in the comparison.

Another possible source of error is the use of body-Jacobian instead of the tool-Jacobian. The change between the two was a translation of 0.05 m in the z-direction and 0.03 m in the negative y-direction in the body frame placing the point of attack for the end effector forces at the top of the handle closest to its ring.

The decision to use the slightly less time-consuming transformation of the geometric Jacobian into the body Jacobian instead of the tool Jacobian did not result in any discernible changes in the forces and torques recorded by the F/T sensor. This way of investigating the influence of the Jacobian does have the same fundamental flaw as the comparison between with and without F/T sensor. However, the change between two theoretical force and torque values multiplied with the body and tool Jacobians were smaller than the uncertainty in the F/T sensor.

It is possible that the change from tool-to body Jacobian would have a noticeable impact if the F/T sensor were better calibrated, but this was not investigated due to time constraints.

It is most likely that the impact of the use of body Jacobian instead of tool Jacobian combined with the relatively basic calibration of the F/T sensor has less impact than the low frequency of the regulation.

The real and the perfect robot

One of the major advantages of this implementation is the virtual robot. Although a simple zero-order hold used for integration it has many advantages over a real robot. The identified model and the matrices created from this identified model in the regulator are the same as the internal matrices used in its simulation, no internal regulator is turning the torque command into current and there is no unidentified friction in the joints that can result in errors.

In contrast to the real robot, there are potential error sources in the identification of its model resulting in potential errors in the mass, Jacobean and Coriolis matrix created from this model. Moreover, there is a possibility of higher friction than compensated for. There are also errors in the implementation to be taken into account. The identification program used by Cognibotics does not estimate the Coulomb friction, and it was approximated by increasing the torque until the joint started to move.

Nevertheless based on the tests done on the implemented algorithm and its response it is clear that the implementation works and performs its function. These errors have less an impact than the low frequency of the regulation.

8

Conclusion

This project sets out two main objectives, the creation of a reliable haptic interface and enhancing the user experience.

Starting with performance, singularity free operation was achieved using damped pseudo-inverse of Γ as calculated in Equation (3.32) and tested in the singular configuration seen in Figure 7.4. Based on the error in Figure 7.2 and the motions of the TCP for both robots in Figure 7.1 singularity free motion tracking in free space have been achieved.

Free-space motion where position and orientation offsets are retained was achieved and investigated during the free space tracking test. The error was larger than hoped for, but it returned to zero within 5 seconds. Moreover, consistent and intuitive results of input signals were achieved and tested during the surface collision test. The controller remained stable during both free-space and constraint motion for all five iterations of tests.

As for the user experience where the ability for the user to set position and orientation offsets without entering into the code was achieved by having the position and rotation offset set automatically between the starting positions of the two robots. This resulted that the position and rotation offset could be set by the user by simply moving the two robot arms to the intended configuration before starting the program without ever having to change any internal code.

This implementation was intended for two real robots resulting in that the virtual robot's starting configuration was decided by using the real robot as a handle. For the test were the real and virtual robots started from different configuration were the start configuration of the virtual robot given in the code and not inferred from the configuration of the real robot.

The feeling of force upon interaction with an object was achieved and tested during free-space tracking, surface collision and physical limitation test under slow motions, however, it might not reach the goal of a realistic and intuitive feeling of force.

This is probably due to two factors, the limitations of friction in the joints of the real robot and the low regulator frequency.

The friction of the joint would hamper the expression of smaller forces and torques experienced by the virtual robot. This was lessened by the inclusion of a direction based offset on the torque input to the real robots joints in an attempt to make the real robot more sensitive to the small imputes from the algorithm. This did not solve the entire problem with the uncompensated factor of friction in the joints of the real robot only lessened some of its resulting flaws.

The low frequency of the regulator thread resulted in that during fast changes in direction the virtual arm continued in the previous direction, something that manifested itself for the real robot arm as an unexpected resistance to movement in the new direction.

This problem was mitigated by lowering the mass of the virtual robot but was still noticeable at faster movements. However, as these faster movements were in the upper end of fast movements what was expected for teaching this is not considered a defining problem.

The last part regarded the implementation of a workable teaching handle to improve adjustments of the slave arm that was implemented.

The project is considered complete as the objectives of the project were achieved in regard to the time allotted.

8.1 Future work

Future work with this implementation of the haptic algorithm will continue on the reduction of calculation time in hope of minimizing the delay in response from the virtual robot to movement from the real robot. Suggested ways that this might be done is to continue tailoring the code for calculation of the geometric Jacobian, mass, Coriolis and centripetal matrices.

A meticulous investigation into the delay between robot and computer would help in locating the source of the bottlenecks. Further testing would include reversing the master, slave relationship of the real and virtual robot to create identical input signals for investigation of the influence of the F/T sensor. A more responsive visualization of the virtual robot would also improve testing.

Bibliography

Åström, K. and B. Wittenmark (2011). *Computer-controlled systems theory and design third edition*. Dover Publications Inc, p. 36. ISBN: 9780486486130.

Bargiel, S., K. Rabenorosoa, C. Clévy, C. Gorecki, and P. Lutz (2010). "Towards micro-assembly of hybrid MOEMS components on a reconfigurable silicon free-space micro-optical bench". *Journal of Micromechanics and Microengineering* **20**:4, p. 045012. DOI: 10.1088/0960-1317/20/4/045012. URL: https://doi.org/10.1088/0960-1317/20/4/045012.

Bloch, A. (2003). *Nonholonomic Mechanics and Control*. Interdisciplinary Applied Mathematics. Springer-Verlag New York. ISBN: 9780387216447.

Burdick, J. (2020). *Damped pseudo inverse*. URL: http://robotics.caltech.edu/~jwb/courses/ME115/handouts/damped.pdf. Accessed: 2020-12-25.

Chen, Y., J. Zhang, C. Yang, and B. Niu (2007). "The workspace mapping with deficient-dof space for the Puma 560 robot and its exoskeleton arm by using orthogonal experiment design method". *Robotics and Computer-Integrated Manufacturing* **23**:4, pp. 478–487. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2006.05.007.

Denavit–Hartenberg parameters (2020). URL: https://www.universal-robots.com/articles/ur/parameters-for-calculations-of-kinematics-and-dynamics/. Accessed: 2020-11-07.

FMU (2020). URL: https://fmi-standard.org/. Accessed: 2020-12-05.

Ghazaei Ardakani, M., M. Karlsson, K. Nilsson, A. Robertsson, and R. Johansson (2018). "Master-slave coordination using virtual constraints for a redundant dual-arm haptic interface". 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems; Conference date: October 2018, pp. 8751–8757. DOI: 10.1109/IROS.2018.8594260.

"History of Industrial Robots" (2012). *International Federation of Robotics*. DOI: https://web.archive.org/web/20121224213437/http://www.ifr.org/uploads/media/History_of_Industrial_Robots_online_brochure_by_IFR_2012.pdf. Accessed: 2020-06-29.

ILM for Basic Modeling and Identification (2020). URL: https://arm.ual.es/ilm/modeling.php. Accessed: 2020-12-20.

Johansson, R. (2019). *Predictive and Adaptive Control Lecture Notes FRTN15 Predictive Control*. Lund University, Dept. Automatic Control, p. 359.

Lee, D. and P. Li (2005). "Passive bilateral control and tool dynamics rendering for nonlinear mechanical teleoperators". *IEEE Transactions on Robotics* **21**:5, pp. 936–951. DOI: 10.1109/TRO.2005.852259.

Lynch, K. M. and F. C. Park (2019a). *Modern robotics mechanics, planning and control*. Cambridge University Press, 67, eq 3.191. ISBN: 9781107156302.

Lynch, K. M. and F. C. Park (2019b). *Modern robotics mechanics, planning and control*. Cambridge University Press, p. 270. ISBN: 9781107156302.

Lynch, K. M. and F. C. Park (2019c). *Modern robotics mechanics, planning and control*. Cambridge University Press, 583, eq B.10–B.11. ISBN: 9781107156302.

Lynch, K. M. and F. C. Park (2019d). *Modern robotics mechanics, planning and control*. Cambridge University Press, 87, eq 3.62. ISBN: 9781107156302.

Lynch, K. M. and F. C. Park (2019e). *Modern robotics mechanics, planning and control*. Cambridge University Press, 98, def 3.20. ISBN: 9781107156302.

Maple (2020). URL: https://www.maplesoft.com/products/maple/professional/. Accessed: 2020-09-11.

Maplesim (2020). URL: https://www.maplesoft.com/products/maplesim/. Accessed: 2020-09-11.

optoforce (2017). *Optoforce HEX-E v1 F/T sensor datasheet*.

OptoForce General DAQ - protocol description 1.4 - USB, CAN, UART (2015).

PyFMI (2020). URL: https://pypi.org/project/PyFMI/. Accessed: 2020-12-05.

RTDE (2020). URL: https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/. Accessed: 2020-10-10.

Sang Hyoung, L., S. Il Hong, C. Sylvain, and R. Johansson (2015). "Autonomous framework for segmenting robot trajectories of manipulation task". *Auton Robot* 38, pp. 107–141. DOI: https://doi.org/10.1007/s10514-014-9397-9.

Siciliano, B., L. Sciavicco, L. Villani, and G. Oriolo (2009). *Robotics Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer-Verlag London Limited, p. 113. ISBN: 9781846286414.

UR5e Information (2020). URL: https://www.universal-robots.com/media/1802778/ur5e-32528_ur_technical_details_.pdf. Accessed: 2020-10-15.

A

Dimensions of UR5e robot

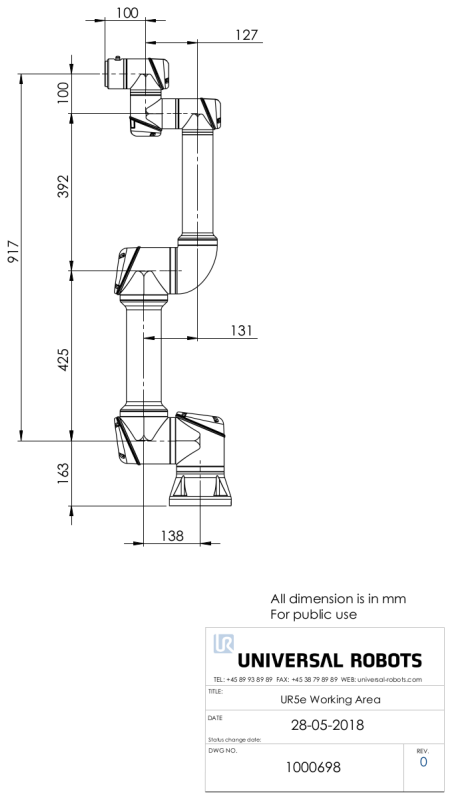


Figure A.1 The dimensions of the UR5e robot used for the creation of the Maplesim virtual robot.

B

Maplesim simulation

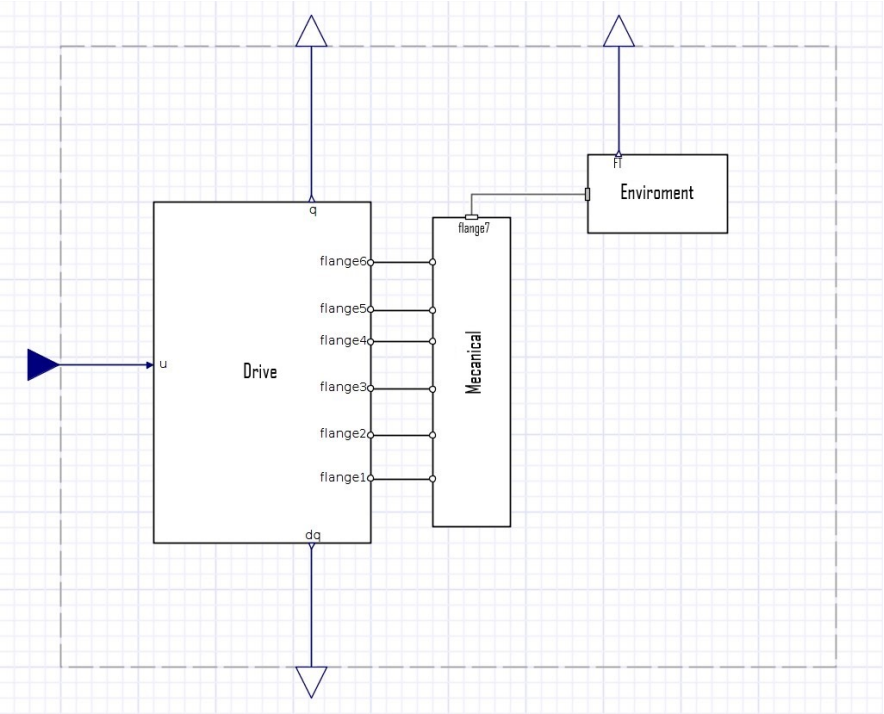


Figure B.1 The overview of the Maplesim model used for the virtual robot. The input torque is given by u to the drive part where it is transformed into a torque around flange 1-6, the joint angle q and the joint angular velocity dq are measured on these flanges. These flanges are connected to each of the six joints in the mechanical part that simulates the kinematics and dynamics of the robot. Flange 7 is the flange of the end effector and is connected to the environment and it is its interaction that results in the forces and torques in the output FT.

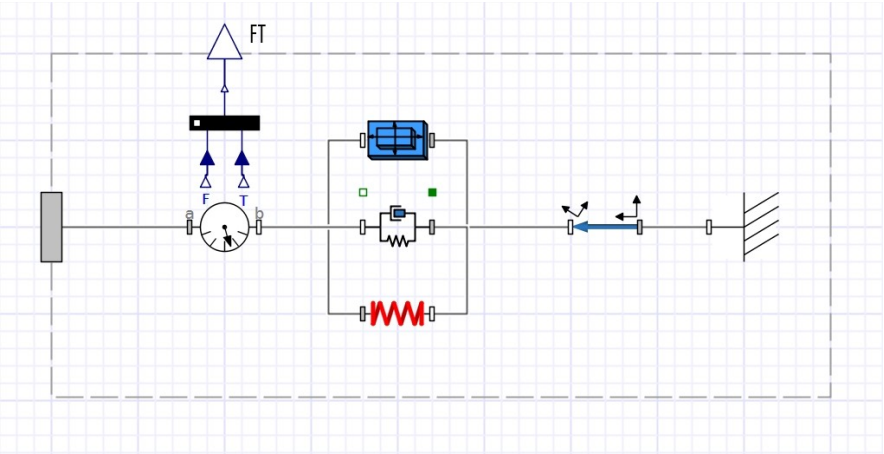


Figure B.2 The insides of the environment part. It is made up of a dampened spring connoted to the ground. Between the spring and the end effector of the simulated robot is the force/torque sensor placed.

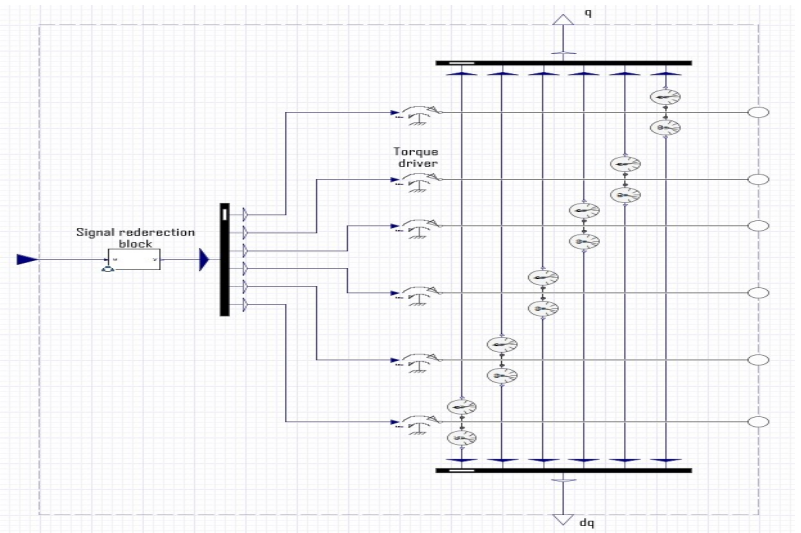


Figure B.3 The insides of the drive part. The input torque is first transposed so that the first value of six in the torque vector is provided to the first joint using the Signal redirection block. The torque is then turned from a real signal into torque using torque drivers, the joint angle and joint velocity are measured on these axes before exiting the drive part.

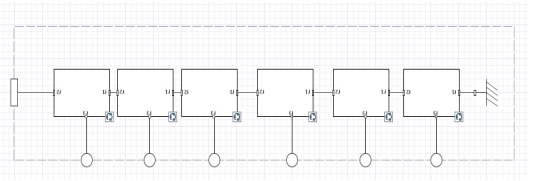
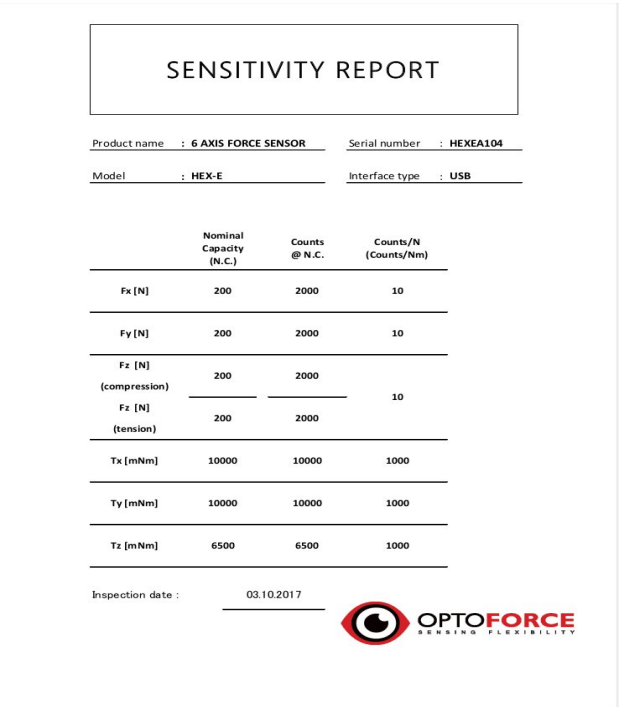


Figure B.4 The insides of the mechanical part. It is made up of six Denavit-Hartenberg subsystems each handling one of the joints. The base of the robot is connected to the ground and the end effector of the robot is given its own flange for connection to the environment part.

C

Optoforce HEX-E v1 sensitivity report



SENSITIVITY REPORT

Product name : 6 AXIS FORCE SENSOR
Serial number : HEXEA104
Model : HEX-E
Interface type : USB

	Nominal Capacity (N.C.)	Counts @ N.C.	Counts/N (Counts/Nm)
Fx [N]	200	2000	10
Fy [N]	200	2000	10
Fz [N] (compression)	200	2000	10
Fz [N] (tension)	200	2000	
Tx [mNm]	10000	10000	1000
Ty [mNm]	10000	10000	1000
Tz [mNm]	6500	6500	1000

Inspection date : 03.10.2017

OPTOFORCE
SENSING FLEXIBILITY

Figure C.1 The sensitivity report of the Optoforce HEX-E v1 F/T sensor (serial number HEXEA104).

D

Assigned coordinate frames

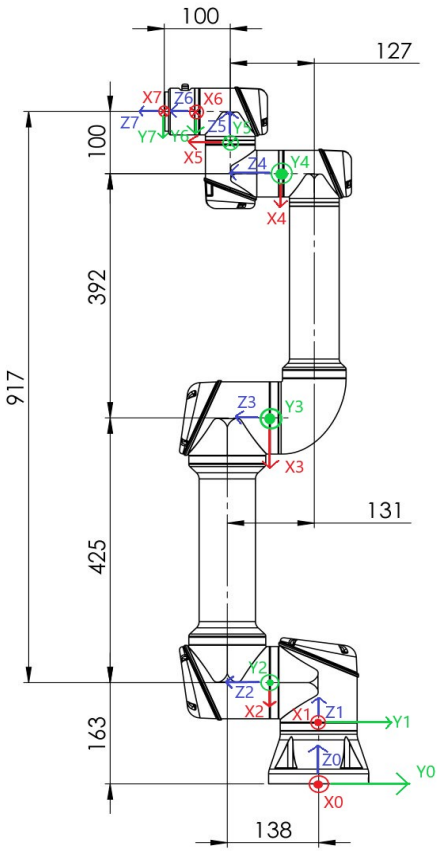


Figure D.1 The coordinate frames used for calculating the homogeneous transformation matrices

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

Document name
MASTER'S THESIS

Date of issue
February 2022

Document Number
TFRT-6153

Author(s)
Eric Ragnarsson

Supervisor
Mahdi Ghazaei, Cognibotics,, Sweden
Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden
Karl- Erik Årzén, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Advanced Teleoperation with Haptic Feedback

Abstract

As the world moves more and more towards automation of repetitive and dangerous tasks there are still many tasks that primarily involve interaction with an object that are still carried out by humans. This is in many ways due to difficulties in programming the robot to achieve similar performance. The most natural way of transferring these skills to the robot is by demonstration. This can be achieved by the use of haptic feedback to allow the operator to feel the robotic system during the demonstration.

This master thesis seeks to implement and improve master-slave coordination using virtual constraints for a redundant dual-arm haptic interface. This is done by implementing a general-purpose algorithm compatible with different types of 6 degrees of freedom robots in Python. For this project a UR5e robot was used as well as a second virtual clone of the UR5e robot to serve as master and slave arm respectively.

The virtual second UR5e robot was simulated using both Maplesim and inverse dynamics. The inclusion of force-torque (F/T) sensors was implemented and tested. Free-space motion where position and orientation offsets are retained as well as collision with a soft surface resulting in a feeling of force upon interaction with a soft object were achieved. The singularity-free operation produced consistent and intuitive results, with the inclusion of a teaching handle to improve adjustments. However, low sampling frequency of the implemented algorithm resulted in delays that negatively impacted the translational and rotational error between the end effectors of these robots, as well as not reaching a realistic and intuitive feeling of force upon interaction with a soft object. These problems were not solved due to lack of time.

The impact of the implemented F/T sensor was investigated, but due to the limited time left was this not enough to reach a definitive conclusion.

The implemented algorithm functions well during slower movements, with room for improvements in the optimization of calculation times and further investigation of the impact of the implemented F/T sensor.

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