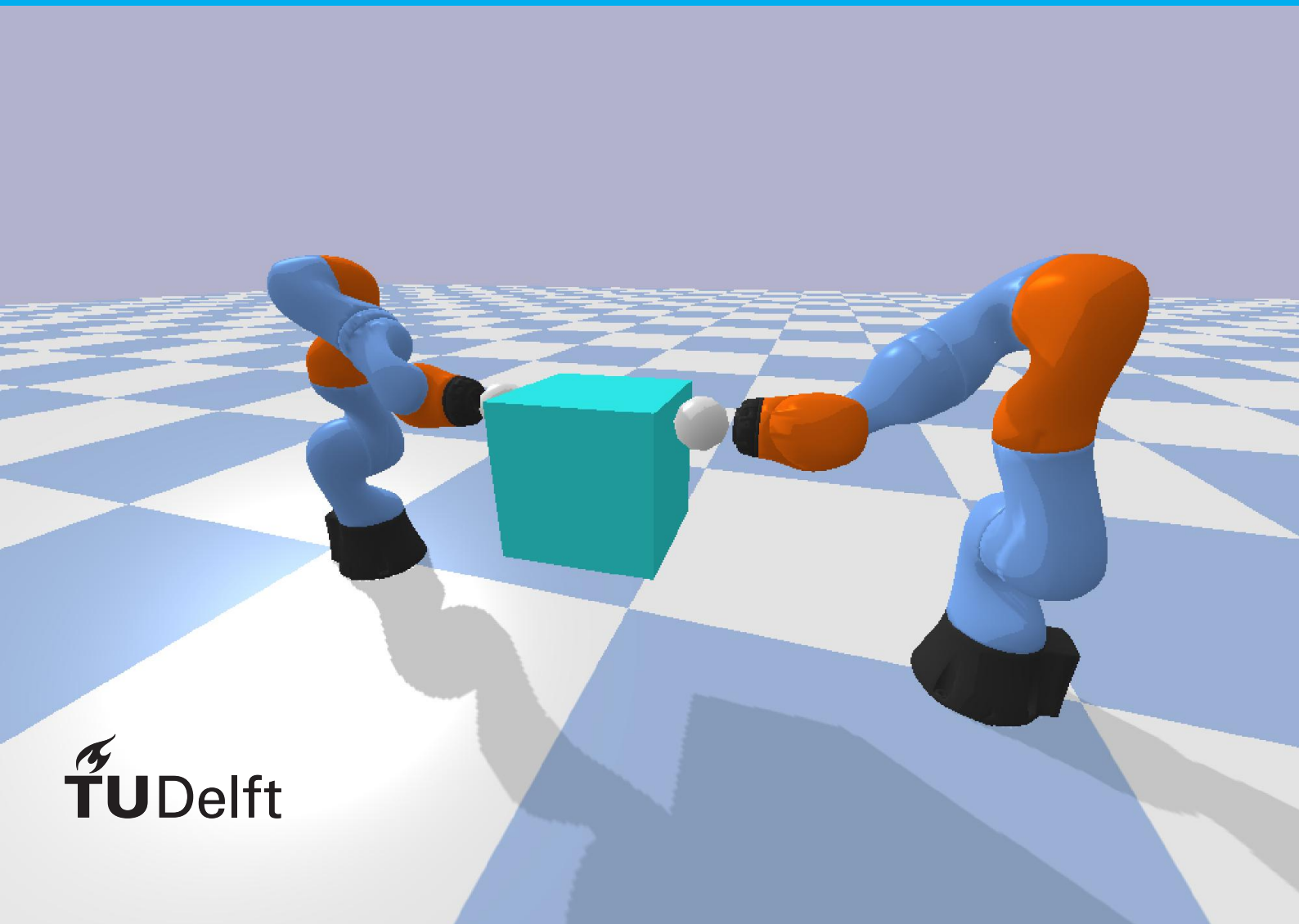
Cooperative Robot Manipulators for Parcel picking and placing

Olav Jacobs

Final version: May 15, 2020



TU Delft

Cooperative Robot Manipulators for Parcel picking and placing

by

Olav Jacobs

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday May 29, 2020 at 1:00 PM.

Student number:	4305213	
Project duration:	June 1, 2019 – May 15, 2020	
Thesis committee:	Dr. ir. J. Kober,	TU Delft, supervisor
	Ir. J. Damsteeg	PrimeVision, supervisor
	Dr. B. Shyrokau	TU Delft
	Ir. G. Franzese	TU Delft

TU Delft

Abstract

In this thesis, a control scheme for lifting parcels using two robot manipulators is presented. The robots do not have a rigid grasp on the object. Instead, they use friction to lift the parcels.

First, a controller calculates the desired force to make sure the parcels do not slip. The required force, as well as a trajectory are then sent to a hybrid force position controller, which controls the force the robot is pushing with in the direction perpendicular to the contact surface, while controlling the position in all other directions. The control scheme thus controls both robots separately given a desired force and position. That way the controller a lot simpler compared to closed kinematic chain controllers, which aim to control the entire system as a single robot.

Since the parcels weight is unknown for an actual system where the robots pick and place parcels, the controller is extended with an adaptive part, that estimates the weight of the parcel. The estimated weight can than be used to improve the feedfordward controller, making the system less reliant on feedback. The estimator works by calculating the difference between the predicted forward dynamics and the actual dynamics of the system. It is shown that the estimator yields good results for estimating the mass of parcels. Furthermore, the trajectory tracking property of the system is improved when using the estimated weight for feedforward control.

Contents

1 Introduction

This thesis is preformed in cooperation with PrimeVision. PrimeVision is an company that develops software solutions for the postal industry. Recently, they have been expanding their business to develop robots for making sorting facilities more efficient. This thesis is part PrimeVisions future developments to further improve the efficiency of these facilities.

In the parcel industry, parcels that come in at a sorting facility need to be placed on the conveyor system in order to be sorted. At this time, moving these parcels is almost always done by hand. PrimeVision is interested to investigate the possibilities for automating this process, with the use of robot manipulators. A picture of the current situation in sorting centers can be seen in Figure 1.1.

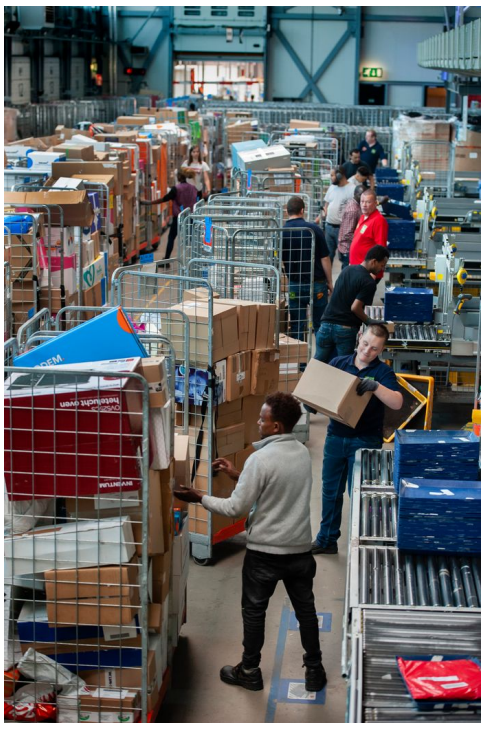


Figure 1.1: Current situation in sorting center [6]

The standard approach for lifting an object with a robot manipulator would be to design a suitable gripper. The difficulty with this project however, is that the parcels do not come in predetermined shapes, sizes and materials. Therefore designing grippers that would be able to lift all parcels, may be very difficult, and the

grippers would most likely be costly to make, due to the many moving parts and actuators. Most likely these grippers will still be limited to a specific range of parcels or materials. Therefore the lifting of a parcel with two cooperative robot manipulators is investigated. By lifting the parcel with two manipulators, the robots will be able to lift a greater variety of parcels, without the need of complicated gripper design. The inspiration for this approach mainly comes from the way a human would lift a larger parcel. When using only one arm, lifting a parcel can be very difficult, but when using two arm, it becomes a lot easier. An image of a simulation of the proposed system is shown in Figure 1.2.

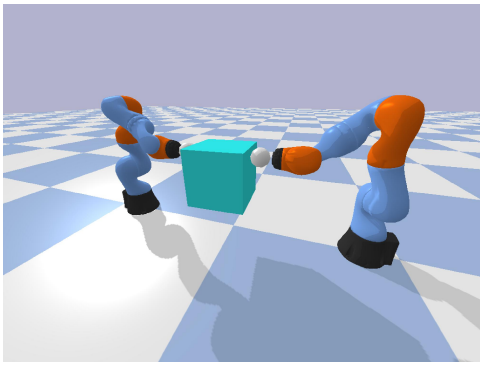


Figure 1.2: Picture of a simulation with two robot manipulators lifting an object

The research question for this Thesis is formulated as: "What is a suitable control scheme for two robot manipulators lifting an object with unknown mass?" With this research question, two main goals for the controller are presented:

The first goal is that the system should have a very low failure rate, as any failure in the system would most likely mean that the system has to be shut down, and someone has to come in and remove the dropped parcel. Furthermore, failure could mean that the parcels get damaged.

The second goal is that the controller should be able to track fast trajectories, in order to be able to move a large amount of parcels. This is especially important as sorting centers have to sort a lot of parcels in a limited amount of time. When compared to the current situation, where humans move the parcels on the conveyor, the robots should not be much slower. For this thesis, the aim is to achieve 5 second operation cycles, which means the robots should be able to pick and place a parcel in about 2 to 3 seconds. The 5 seconds operation cycles are chosen to have a fast but plausible goal for the speed of the system.

In this thesis, a control scheme is represented for lifting a parcel using two robots, where the robots do not have a rigid grasp on the object, but instead use friction in order to prevent the parcel from slipping. Because the weight of parcel is unknown in a real scenario, the controller is extended with an adaptive part, that estimates the weight of parcels as they are being moved. The adaptive controller reduces the amount of feedback required for manipulation, and the system achieves better trajectory tracking when using the weight estimation for feedforward control. The controller is tested using a simulation, which shows that the controller can achieve good results for parcel up to a certain weight.

I would like to thank PrimeVision for the opportunity to do this thesis at their company. I would also like to thank Jens Kober and Jan-Willem Damsteeg for their guidance during each stage of this project.

2

Related work

In this chapter, a short overview of relevant literature will be presented. The literature is about multiple robots controlling an object. There are plenty of works available which aim to control objects using multiple robot manipulators. At the end of this Chapter a motivation for this thesis is presented, by showing the difference between this work and previous papers.

This chapter is divided in three sections. The Section 2.1 is about more traditional control methods. These methods mostly use algebraic formulas to calculate a desired control input. The Section 2.2 is about impedance controllers. Impedance controllers give a set of dynamic parameters which the robot should follow, essentially making the robot mimic a system with these parameters. The Section 2.3 is about adaptive control schemes and learning controllers. Finally, in Section 2.4, the different control methods are compared to each other, and the controller presented in this paper.

2.1. Traditional control methods

A paper about the controller for two manipulators lifting an object is [15]. In this paper a control scheme is designed for position tracking of an object between two robot arms with rolling contacts. The paper expands on this by giving a method to control position of the rolling contacts on the object. The method always keeps track of the necessary pressure to maintain a grip without slipping, while also making sure the forces on the object do not get too big, which could damage the object, or the manipulators could reach the end of their capacity. The experimental results leave some room for improvement, but the method forms a good starting point for controlling an object with dual manipulators.

For the control of an object between two robot fingers, [14] is an interesting paper to study, since they are able to control the position of a object between two fingers, without object sensing. This makes it a lot more useful for our application, since any control method that would require some kind of sensing of the object would make the system a lot more complex and expensive. The method only requires the robots kinematics and the robots states. The paper is however limited to the planar case, and the object needs to have parallel rigid surfaces. The paper gives us the control inputs necessary to control the position and orientation of the object (if the system is not redundant), while keeping the system stable. The mathematical conclusions are than validated with an experiment. The setup of the experiment can be seen in Figure 2.1.

A similar control method is described in [3]. The difference is that the control method described in this paper also works for non-parallel surfaces. The method should be able to find stable control with two fingers on any convex polygon shape. Contrary to [14] however it does require estimates of the kinematic parameters of the object to get a stable grip. Both papers limit themselves to the 2D case and both prove the stability of their method using manifold theory [4], as it saves them from having to calculate the inverse kinematic of the robotic fingers.

An other paper about the control of an object with two robot fingers is [7]. In this paper, the control of a compliant object between two robot fingers is investigated. This should be relevant since most parcels will

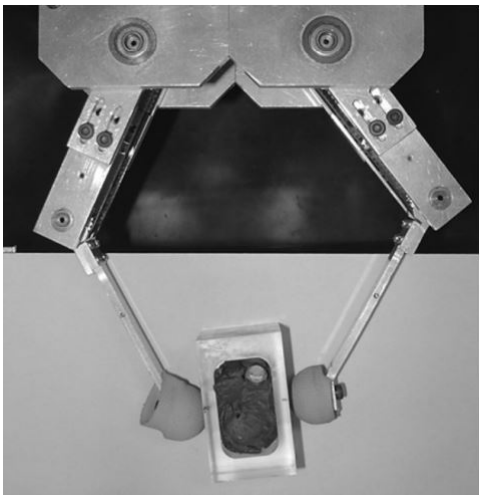


Figure 2.1: Experimental setup for the control of an object between two fingers [14]

be somewhat compliant as well. The methods works fairly similar to the previous papers mentioned, by calculating the dynamics of the robot and the object, and finding a control scheme to control the position and orientation of the object while keeping the force on the object high enough for the friction to keep the object from sliding. The compliance in the object makes the problem a bit more difficult, since there are more independent states in the system, and the compliance causes some singularities in the system matrices. The paper explains how to solve these singularities by transformations of the output. The proposed method is validated with a simulation. The method is also limited to the planar case.

An other part of the control of the parcels, is the monitoring of the internal forces and moments. This is important, since the parcels can only be subject to a certain amount of force, or they will get damaged. A method for calculating internal forces of an object being manipulated by multiple robots is given in [17]. The paper describes using virtual linkages between the contact points to calculate the internal forces and moments. The approach could however be too complex, since control methods like the one proposed in [14] already allow for the control of the force the robot is using to lift the parcel, which might be sufficient to not damage the parcels.

2.2. Impedance control

In [11], a impedance control method is proposed, with use of a relative Jacobian. The advantage of using a relative Jacobian is that it will be more intuitive to program bimanual tasks, such as assembly, since the relative motion between the two manipulators can be controlled directly. The impedance controller allows for the control of the way the robot arms interact with their environment, by designing the a stiffness and dampening matrix. Essentially the controller is designed in such a way that the closed loop system mimics the physical attributes of these two matrices. This is very useful for robots that work in an environment together with humans, since it allows for reduction of force that the robots use, making is safer for humans to get close or even interact with the robots. The paper does however focus on other bimanual tasks such as assembly, and not so much on lifting objects. Therefore it will not be very useful for this thesis

Object impedance control [16] can also be implemented. In this case the physical attributes of the object are controlled rather than attributes of the robots themselves. This has more upsides for assembly tasks, but it is also useful for controlling the internal forces of the object. The object can be modelled as a simple mass spring system, but more complicated models can be used as well. One of the main advantages of this method is that we only have to plan the path of the object. This avoids having to deal with the high number of degrees of freedom of the robots. Impedance control also gives an easy way to do obstacle avoidance, as the method allows for the modelling of obstacles as potential fields. The object impedance controller does require force sensors at the manipulators, or the implementation will be a lot more complex and also require detailed knowledge about the objects dynamic parameters. The controller is tested and compared to some other controllers via an experiment, which shows the object impedance controller superiority in inertia compensation

and dealing with external forces.

2.3. Learning algorithms

An adaptative control scheme is proposed in [2]. The method is able to compensate for uncertainties of the kinematic parameters of a closed kinematic chain. It uses the velocity and position reading of two manipulators lifting a common object to estimate the kinematic parameters of the object and the base of the robots, so there is no need for manipulator force sensors or visual feedback. It than uses a the inverse dynamics for the design of the controllers. Stability of the controller is proven using ultimate boundedness. In a physical experiment it is shown that the adaptive controller can indeed achieve very good reference tracking when compared to a non-adaptive controller. In this paper it is assumed that the object lifted is a rigid body, and that all grasps on the object are rigid. For stability it is also assumed that the initial guess for the kinematics of the closed chain are within a certain range around the true parameters.

Neural network controllers have also been investigated for bimanual robots lifting an object [18]. The neural network controller has the ability to be robust when dealing with uncertainties in the robots dynamics or the objects parameters. This makes it especially useful for unstructured environments. The method works by first calculating the kinematics of the system, and specifying the requirements. Then a barrier Lyapunov function is designed. This function gives a barrier for the tracking of a reference. The Lyapunov function is then used for the synthesis of the neural network controller. To achieve global stability, a second neural network is introduced, which is synthesized from a regular global Lyapunov function. The final controller switches between the two neural networks, to ensure global stability. A bimanual robot is used in a physical experiment which show the effectiveness of the controller.

Adaptive fuzzy logic controllers have also been used for the manipulation of an object with multiple robot arms [9, 10]. Adaptive fuzzy logic controllers can be made to be robust when dealing with model uncertainties. This is useful when the object being lifted is not very well defined. Stability is proven in both papers by Lyapunov theorem. Both of the papers use decentralized control to contain the computational size of the problem. Even though the fuzzy logic method eliminates the need to calculate the inverse kinematics of the robots, the method needs a lot of rules, even after applying a fuzzy rule reduction algorithm, so in terms of computational requirements, fuzzy logic is not very efficient. In the second paper, a simple formula is added to the system to deal with the robots motors deadzone (the zone in which the power of the motors is too low to overcome the friction).

In [8] a method using dynamic movement primitives (DMP) is proposed. Such a system works by constructing a (non-linear) differential equation for certain movement. Changing some of the parameters of this equation can change the dynamics of the movement. Since some move has to be pre-defined, DMP is often used in an imitation learning algorithm. In this paper the DMP is updated after each movement using an iterative learning control. This method constructs a function which is a sum of radial basis functions called a modulation term, which is added to the differential equation of the system. While DMP is often used to solve kinematics of a robotic arm, this paper controls the robots force/torque, and secondly, the paper describes a method to couple two robot arms. This has the advantage that the method can now be used for bimanual tasks which require a specific amount of force, such as bimanual lifting. The main advantage of using a DMP method is that planning for obstacle avoidance, including the two robots avoiding each other is easy to implement, since the robot can learn it online from its own sensory feedback (vision and force). The results of the paper are verified by multiple experiments using two robot arms preforming a bimanual task. The setup is shown in Figure 2.2.

2.4. Discussion

In this section the control methods above will be compared to see their differences. Furthermore, the decision to make a new control scheme in this thesis will be motivated.

Most of the papers in this chapter are in Table 2.1 with information about the controller used in that paper.

From the table it can be derived which control scheme would be most suited for the problem of this project.

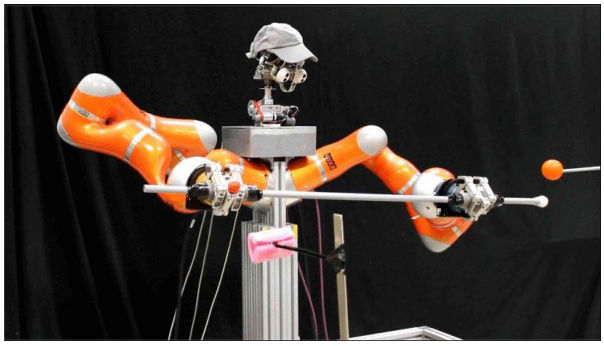


Figure 2.2: Bimanual robot used for the experiments of [8]

Paper	Year	Control method	Grasp	Dimensions	Other notes
E. Paljug: Control of rolling contacts in multi-arm manipulation [15]	1994	Linear feedback/pole placement	Rolling contacts	Planar	Also able to control the position of the contacts on the object
R. Ozawa: Control of an object with parallel surfaces by a pair of finger robots without object sensing [14]	2005	Linear feedback	Rolling contacts	Planar	Does not require inverse kinematics or inverse dynamics, also does not require the objects kinematics or dynamics
S. Arimoto: Dynamic force/torque balance of 2D polygonal objects by a pair of rolling contacts and sensory-motor coordination [3]	2003	Non-linear feedback	Rolling contacts	Planar	Control of object with non-parallel surfaces, requires extra sensors for orientation control
Z. Doulgeri: Nonlinear manipulation control of a compliant object by dual fingers [7]	2006	Linear feedback	Point contact with friction	Planar	Considers compliant object instead of a rigid body
J. Lee: Relative impedance control for dual-arm robots performing asymmetric bimanual tasks [11]	2013	Impedance control	-	3D	Paper focusses on robots preforming a bimanual task instead of manipulating an object
S. Schneider: Object impedance control for cooperative manipulation: Theory and experimental results [16]	1992	Object Impedance control	Rigid	Planar	Impedance controller requires detailed knowledge about objects dynamics
F. Aghili: Adaptive control of manipulators forming closed kinematic chain with inaccurate kinematic model [2]	2012	Inverse dynamics controller with estimator for adaptive kinematics	Rigid	3D	The rigid links make finding the missing kinematic parameters a lot easier that point or rolling contacts would
C. Yang: Neural control of bimanual robots with guaranteed global stability and motion precision [18]	2016	Adaptive controller based on radial basis function neural network	Rigid	3D	
W. Gueaieb: A robust adaptive fuzzy position/force control scheme for cooperative manipulators [9]	2003	Adaptive controller based on fuzzy logic	Rigid	3D	
A. Gams: Coupling movement primitives: Interaction with the environment and bimanual tasks [8]	2014	Dynamic movement primitives	Rigid	3D	

Table 2.1: Comparison table some papers using different controllers for bimanual lifting

It can be seen that a lot of the more complex control scheme such a impedance [16] or adaptive controllers [2, 8, 9, 18], usually assume there is a rigid grasp on the object. That way, the problem really only has one objective, the tracking of a reference. Therefore the problem reduces to a closed chain robot controller.

When not assuming a rigid grasp of the object, the controller must comply to multiple objectives. The robot now has to keep a certain amount of force under the right angle on the object at all times, in order to maintain force closure. However some of these control schemes assume a planar case of two finger robots [3, 7, 14, 15]. This can become a problem when working with rolling contacts, as the rolling constraint is homonymic in the planar case, but it becomes non-homonymic when expanding to a 3 dimensional model.

It can be concluded that for this project, all control methods would need some adjustments to work properly. The rolling finger models will have to be expanded to a 3 dimensional robot. The methods for adaptive controller for bimanual lifting usually assume a rigid grip, so these methods will need to be changed to make them able to maintain force closure on the manipulated object (the concept of force closure is further explained in Section 3.2.2).

Since none of the controller offered in the literature give a simple solution for controlling an object with rolling contacts in a 3D space, a new controller will be devised. Contrary to the literature presented, this controller will be able to move a object along a 3D trajectory while keeping the enough force on the object for it not slip. Since making a controller for a closed chain in 3D is a very complex problem, the two robots will each be controller separately. That makes it a lot easier to control the robots position and force. A detailed explanation of the controller is presented in the Chapter 3.

3

Controller

In this chapter, the controller will be presented. The controller consists of two main parts. The first part is a controller which calculates the forces necessary to constrain the parcel. This is presented in Section 3.2 The second part is a controller which takes the desired forces and position and calculates the desired joint torques, which are sent to the robots actuators. This is presented in Section 3.3 Firstly we will discuss the control architecture of the entire system in Section 3.1

3.1. Controller overview

In this section, an overview of the control architecture will be presented. A schematic of the control architecture is given in Figure 3.1.

The controller takes a trajectory as its input. The trajectory consists of a desired position for the parcel at all t. From this, the desired acceleration and velocity can be derived.

The desired position and acceleration are than send to the first controller. This controller calculates the desired forces on the parcel. The reason that the forces are calculated instead of just using an arbitrarily large force, which will always generate enough friction to lift the parcel, is that the parcels may get damaged when the contact forces are too large. Therefore it is desirable that the minimal required force to lift the parcel is used. This controller will also use the forces on the parcel to control the position of the parcel in the direction the robots are pushing.

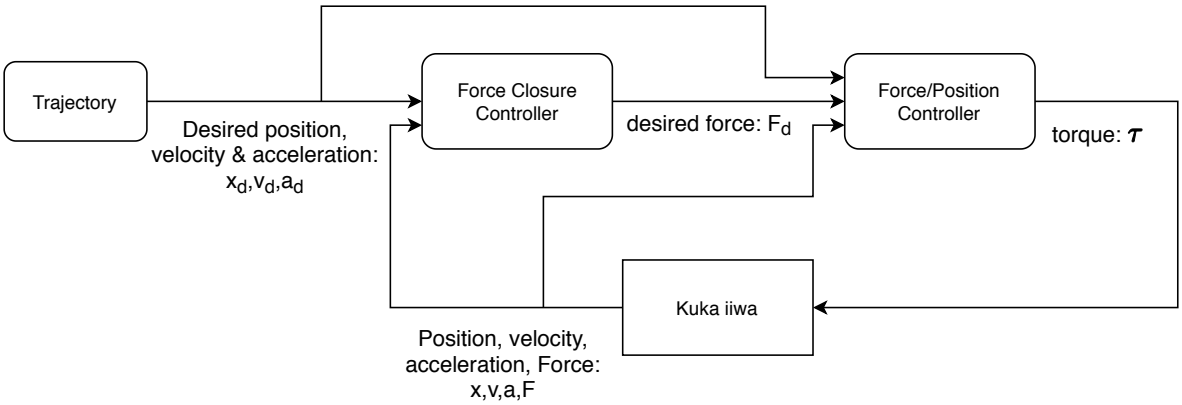


Figure 3.1: A schematic view of the control architecture of the system

The desired force, and positions are then sent to the second controller. This is a hybrid force/position controller, derived from a similar controller which is described in K. Lynch: Modern Robotics [12]. The controller transforms all the dynamics to the robots task space, which is the frame a the robots end effector. This means that we can control the robot directly by giving it the required positions, in Cartesian space, rotate them to

the end effector frame, and than the robot will move there. This avoids the need for inverse kinematics calculations, since the controller essentially already preforms the inverse kinematics internally. The reason we use this transformation, is because we can control each direction in Cartesian space individually. Therefore we can control the position in all but one direction, and control the force the robot is pushing with in an other direction. The controller will be discussed in more detail in Section 3.3.

Since the second controller produces the desired joint torques of each joint, they can be sent to the robots actuator for torque control. The position and contact forces are returned and used by both the controllers for feedback.

3.2. Force closure controller

The first controller is the force closure controller. It calculates the required force to constrain the parcel, and prevent it from slipping. The calculation is based on the external forces, which are gravity and momentum from acceleration or deceleration. The controller uses the Coulomb friction model to calculate the required internal force that the robots need to push with. The controller also calculate the external force in the direction of pushing, to control the position in this direction.

3.2.1. Friction model

The controller is based on the Coulomb friction model. This means that the static friction is assumed to be linear. This means that the maximal static friction that a contact point can have without slipping is linear to the normal force at that contact point. The friction coefficient (μ) gives us the relation between the normal force and the maximal friction force as:

$$f_f = \mu f_n \tag{3.1}$$

Here, f_f is the friction force, and f_n is the normal force. The friction coefficient μ is dependent on the two contact surfaces. For robot manipulation, often a soft fingertip model is used. This means that the contact is not concentrated in one point, but is spread out over a small surface. This allows the contact to also resist rotations around the contract point. The relation between the normal force of the contact and the maximal torsional friction is similar to the directional friction:

$$\tau_f = \mu_t f_n \tag{3.2}$$

Here τ_f is the maximal torsional friction, and μ_t is the torsional friction coefficient.

For robot manipulation we also define a friction cone. A friction cone is a cone that is drawn from a contact point, that illustrates the ratio between the normal force and the friction force. If the total force at the contact point lies within the friction cone, than the contact point will not slip, see figure 3.2. The angle of the friction cone with respect to the contact normal is $\alpha = \tan^{-1}\mu$ [13].

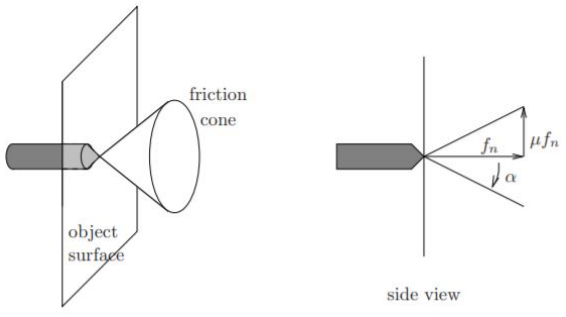


Figure 3.2: Schematic drawing of a friction cone [13]

3.2.2. Force closure

In robot manipulation, force closure is achieved when the contact points on an object can theoretically resist any external forces acting on the object. It is possible to check whether an object is in force closure in 2D, by

drawing the friction cones of the contact points. If we draw a straight line between two contact points, and the line lies within the friction cones of both contact points, than the object is in force closure. The two contact points can resist external forces and moments in all directions, ass long as the normal forces are large enough.

For lifting a simple parcel, with parallel surfaces, this definition of force closure is always achieved, as long as the two contact points are directly opposed to each other (both contact points are on the contact normal of the other contact point). To be able to lift the parcel, it is also necessary to know the actual forces required lift the object.

3.2.3. Internal force

To lift the parcel, we first calculate the required internal force. Using the Coulomb friction model, the required internal forces is linear to the external forces perpendicular to the contacts. We say that the robots push in the x direction. Therefore the external forces in y and z direction need to be calculated. The external force consists of gravity, which only acts in the z direction, and the forces due to accelerations. Thus we formulate the total external force in y and z directions.

$$f = \sqrt{(a_y m)^2 + (a_z m + mg)^2} = m\sqrt{a_y^2 + (a_z + g)^2} \tag{3.3}$$

In this equation, m is the mass of the parcel, a_y and a_z are the accelerations in y and z directions respectively, and g is the gravitational constant. We can easily derive the minimal required internal force by multiplying this force with the friction coefficient. Since this will give us the minimal required force, any error in the model or the calculated forces will cause the parcel to slip. Therefore we also multiply with a safety factor s ($s \geq 1$). So the required internal force is:

$$f_i = \mu s\left(m\sqrt{a_y^2 + (a_z + g)^2}\right) \tag{3.4}$$

Since there are two contact points, both accounting for half the friction, both the manipulators need to push with half of the required internal force.

3.2.4. External force

The hybrid force/position controller presented in Section 3.3 will control the position in all direction, except for the x direction, in which it will control the pushing force of the robot. It is still necessary to control the position of the parcels in the x direction. Therefore, the position in the x direction needs to be controller by the force closure controller. This is done by calculating the desired external force on the parcel. For this a simple feedforward controller is used:

$$f_e = ma_x \tag{3.5}$$

To get rid of any positional errors that might accumulate using only the feedforward controller, a small PD controller is added. No integral term is used, since this normally used to get rid of steady state errors, however, small steady state errors do not matter for this application (small deviations in the position do not matter, as the parcels need to be placed on a conveyor). The required external force becomes:

$$f_e = m\left(a_x + (K_p e + K_d \frac{d}{dt} e)\right) \tag{3.6}$$

Here e is the positional error calculated as $e = x - x_d$. Now that we have the required internal and external forces, we can sum them together to get the total desired force with which the robots should push on the parcels. The forces are summed together as:

$$f_1 = \begin{cases} -0.5 f_i + f_e & \text{if} \quad f_e < 0 \\ -0.5 f_i & \text{if} \quad f_e \geq 0 \end{cases} \tag{3.7}$$

$$f_2 = \begin{cases} 0.5 f_i & \text{if} \quad f_e < 0 \\ 0.5 f_i + f_e & \text{if} \quad f_e \geq 0 \end{cases} \tag{3.8}$$

Since the robots can only push and not pull, the desired external force is summed to one of the two robots, depending on whether the external force is positive or negative.

3.3. Hybrid force position control

Now that the desired force is known, and the desired positions are known from the trajectory, a controller is needed that can control the position in orientation in all directions, except for the x direction. In the x direction the force needs to be controlled. To achieve this, we use the controller from the book Modern Robotics by K. Lynch & F. Park [12]. The controller transforms the robot dynamics to task space (the frame at the end effector of the robot). This allows us to decouple the different directions in Cartesian space. Therefore it is possible to use a different controller for the different directions. In this section, the controller will be explained in some more detail.

3.3.1. Twists

The controller in the book is based on the twist notation. This is an alternative way to formulate rotations and movement in 3D. A twist is defined as follows. Take a transformation matrix $\boldsymbol{T}$, which transforms the world frame to the current body frame, and the spatial velocity $\dot{\boldsymbol{T}}$. $\boldsymbol{T}$ is a concatenation of $\boldsymbol{R}(t)$ (3×3), which is the rotation matrix of the body frame, and $\boldsymbol{p}(t)$ (3×1), which is the position vector of the origin of the body frame. The bottom row of $\boldsymbol{T}$ consists of zeros underneath the rotation matrix, and a 1 underneath the position vector, such that $\boldsymbol{T}_a\boldsymbol{T}_b = \boldsymbol{T}_{ab}$ (multiplying two transformation matrices yields another transformation matrix). That way a frame can be transformed to its position in an other frame. A schematic drawing of a frame transformation is shown in Figure 3.3.

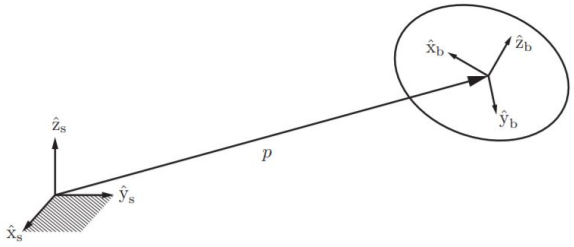


Figure 3.3: Schematic drawing of a frame transformation [12]

$$\boldsymbol{T}(t) = \begin{bmatrix} \boldsymbol{R}(t) & \boldsymbol{p}(t) \\ 0 & 1 \end{bmatrix} \tag{3.9}$$

$$\dot{\boldsymbol{T}}(t) = \begin{bmatrix} \dot{\boldsymbol{R}}(t) & \dot{\boldsymbol{p}}(t) \\ 0 & 0 \end{bmatrix} \tag{3.10}$$

Now, transform the spatial velocity to the body frame.

$$\begin{aligned} \boldsymbol{T}^{-1}\dot{\boldsymbol{T}} &= \begin{bmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T\boldsymbol{p} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{R}} & \dot{\boldsymbol{p}} \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{R}^T\dot{\boldsymbol{R}} & \boldsymbol{R}^T\dot{\boldsymbol{p}} \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} [\boldsymbol{\omega}_b] & \boldsymbol{v}_b \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{3.11}$$

In this notation, $[\boldsymbol{\omega}_b]$ is a skew symmetric matrix representation of the velocity expressed in body coordinates.

$$[\boldsymbol{\omega}_b] = \begin{bmatrix} 0 & -\omega_{b3} & \omega_{b2} \\ \omega_{b3} & 0 & -\omega_{b1} \\ -\omega_{b2} & \omega_{b1} & 0 \end{bmatrix} \tag{3.12}$$

Since $[\boldsymbol{\omega}_b]$ only consists of 3 parameters, it is possible to change it to a vector:

$$\boldsymbol{\omega}_b = \begin{bmatrix} \omega_{b1} \\ \omega_{b2} \\ \omega_{b3} \end{bmatrix} \tag{3.13}$$

This can be combined with the velocity vector $\boldsymbol{v}_b$:

$$\boldsymbol{v}_b = \begin{bmatrix} v_{b1} \\ v_{b2} \\ v_{b3} \end{bmatrix} \tag{3.14}$$

The twist of the object in body frame is now defined as:

$$\mathcal{V}_b = \begin{bmatrix} \boldsymbol{\omega}_b \\ \boldsymbol{v}_b \end{bmatrix} \tag{3.15}$$

For calculations, often the matrix notation of a twist is used.

$$\boldsymbol{T}^{-1}\dot{\boldsymbol{T}} = [\mathcal{V}_b] = \begin{bmatrix} [\boldsymbol{\omega}_b] & \boldsymbol{v}_b \\ 0 & 0 \end{bmatrix} \tag{3.16}$$

3.3.2. Body Jacobian

In robotics, the Jacobian is widely used. The Jacobian transforms the joint velocities to the velocities of the end effector in Cartesian space. In the book, two different kind of Jacobians for robot manipulators are defined. The space Jacobian and the body Jacobian. The space Jacobian is the Jacobian that describes the twist of the end effector in relation to the fixed space frame. The body Jacobian is the Jacobian that describes the end effector twist in relation to the end effector frame. The Jacobians are based on the twists of the robots end effector, instead of the regular Cartesian coordinates. One of the advantages of this notion of the Jacobian, is that the calculation of the Jacobian does not require any derivatives, which results in faster calculations.

The body Jacobian $\boldsymbol{J}_b(\boldsymbol{\theta})$ is defined as follows. It describes the relation between the joint velocities $\dot{\boldsymbol{\theta}}$ and the end effector twist in body frame $\mathcal{V}_b$:

$$\mathcal{V}_b = \boldsymbol{J}_b(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{3.17}$$

The Jacobian at any point in time only depends of the joint positions $\boldsymbol{\theta}$. For readability, for the remainder of this thesis, $\boldsymbol{J}_b(\boldsymbol{\theta})$ will just be written as $\boldsymbol{J}_b$.

3.3.3. Task space dynamics

Using the notion of the body Jacobian, it is possible to transform the regular robot dynamics to task space dynamics. The usual dynamics of a robot manipulator, are as follows:

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{h}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \tag{3.18}$$

Here, $\boldsymbol{\tau}$ is a vector containing the joint torques. $\boldsymbol{M}(\boldsymbol{\theta})$ is the mass matrix of the robot, and $\boldsymbol{h}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ represent the gravity and Coriolis forces. Since the Coriolis forces are quite complex, and the robot moves relatively slow for most trajectories, the Coriolis forces are ignored in this thesis. Therefore the dynamics of the robot are reduced to:

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{g}(\boldsymbol{\theta}) \tag{3.19}$$

Here, $\boldsymbol{g}(\boldsymbol{\theta})$ are the gravitational forces. The gravitational forces are calculated as the change in potential energy due to joint movements:

$$\boldsymbol{g}(\boldsymbol{\theta}) = \frac{\partial P(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{3.20}$$

Here, $P(\boldsymbol{\theta})$ is the total potential energy of the links of the robot. They are simply calculated as $P = \sum_i^n m_i h_i(\boldsymbol{\theta})$, where m_i is the mass of link i, $h_i(\boldsymbol{\theta})$ is the height (or position in z direction) of the center of mass of link i and n is the number of links.

The dynamics can be transformed to the task space. This means that the robots dynamics are described in Cartesian space, rather than in joint space. Since the body Jacobian is used, the dynamics are transformed to

the robots end effector frame. The formula for transforming the robots dynamics to task space are presented below. The full derivation for this transformation can be found in [12].

$$\boldsymbol{\Lambda}(\boldsymbol{\theta}) = \boldsymbol{J}_b^{-T}\boldsymbol{M}(\boldsymbol{\theta})\boldsymbol{J}_b^{-1} \tag{3.21}$$

$$\boldsymbol{\eta}(\theta) = \boldsymbol{J}_b^{-T}\boldsymbol{g}(\boldsymbol{\theta}) \tag{3.22}$$

Since the body Jacobian $\boldsymbol{J}_b(\boldsymbol{\theta})$ needs to be invertible, there needs to be a one to one mapping between the joint velocities and end-effector twists. Therefore this transformation will only be possible if the robot has as many joints as there are velocities in the twist notation. Since there are 6 velocities in twists in 3D, the robot needs to have 6 joints. Furthermore, the Jacobian will not be invertible if the robot is at a singularity configuration. The robot is said to be in a singularity configuration if it is not possible to map all Cartesian velocities to joint velocities. (movement in certain Cartesian direction are not possible from this configuration)

3.3.4. Control scheme

The hybrid controller consists of two controllers. One for the position control and one for the force control. The position controller is as follows:

$$\boldsymbol{\tau} = \boldsymbol{J}_b^T\Big(\boldsymbol{\Lambda}(\boldsymbol{\theta})\Big(\dot{\mathcal{V}}_d + K_p X_e + K_i\int X_e(t)\,dt + K_d\mathcal{V}_e\Big) + \boldsymbol{\eta}(\boldsymbol{\theta})\Big) \tag{3.23}$$

The terms K_p, K_i and K_d are the control parameters for proportional, integral, and derivative control respectively. X_e is the twist that brings the current end effector frame to the desired end effector frame in unit time: $[X_e] = \log([X]^{-1}[X_d])$, where $[X]$ is the matrix notation of the current end effector frame and $[X_d]$ is the matrix notation of the desired end effector frame.

This controller is slightly simplified compared to the controller from the Modern Robotics book [12]. In the book, the desired end effector twist $\mathcal{V}_d$ is transformed to the actual end effector frame. However, this is only relevant when there is a large difference between the desired end effector frame and the actual end effector frame. This should not occur during normal operations. Furthermore, in the book, the last term is for both gravity en Coriolis compensation. However, as stated, Coriolis forces are neglected for this controller.

The force controller looks quite similar to the position controller:

$$\boldsymbol{\tau} = \boldsymbol{g}(\boldsymbol{\theta}) + \boldsymbol{J}_b^T\Big(\mathcal{F}_d + K_{fp}\mathcal{F}_e + K_{fi}\int \mathcal{F}_e(t)\,dt\Big) \tag{3.24}$$

In this controller, $\boldsymbol{g}(\boldsymbol{\theta})$ is the gravity, $\mathcal{F}_d$ is the desired wrench at the end effector. K_{fp} and K_{fi} are the proportional and integral controller gains respectively. Derivative control is not implemented, since the force can change rapidly, and measurements of the force are often noisy, making the derivative unstable. Furthermore, since the forces are directly applied, and do not depend on past values of the control input, the system for the force control is memory-less. In a memory-less system, the control inputs are not integrated by the system. Therefore derivative control is not necessary.

Lastly, to combine the two controller, we need to be able to separate the directions in which the controllers act. To do this, first define the directions in which the robots need to push as constraints. The constraint matrix $\boldsymbol{A}$ can depend on the robots joints positions $\boldsymbol{\theta}$ depending on the type of constraints, however, since the robots always need to push in the x direction, $\boldsymbol{A}$ does not depend on the joint positions. The constraint matrix is defined as follows:

$$\boldsymbol{A}\mathcal{V}_b = 0 \tag{3.25}$$

Since A should constrain the x direction, it becomes:

$$\boldsymbol{A}_s = \begin{bmatrix}0 & 0 & 0 & 1 & 0 & 0\end{bmatrix} \tag{3.26}$$

Since $\boldsymbol{A}$ needs to be in the robots task space, it ,becomes:

$$\boldsymbol{A}_b = \Big(\begin{bmatrix}\boldsymbol{R}_{sb} & 0\\ 0 & \boldsymbol{R}_{sb}\end{bmatrix}\boldsymbol{A}^T\Big)^T \tag{3.27}$$

Where $\boldsymbol{R}_{sb}$ is the rotation matrix from the space frame to the end effector frame.

With $\boldsymbol{A}$ and the task space dynamics $\boldsymbol{\Lambda}(\boldsymbol{\theta})$, a matrix $\boldsymbol{P}(\boldsymbol{\theta})$ can be constructed, which projects a wrench onto subspace of wrenches that move the end effector tangent to the constraints. Similarly, $\boldsymbol{P}(\boldsymbol{\theta}) - \boldsymbol{I}$ projects a wrench onto a subspace of wrenches that act against the constraints.

$$\boldsymbol{P}(\boldsymbol{\theta}) = \boldsymbol{I} - \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{\Lambda}(\boldsymbol{\theta})^{-1}\boldsymbol{A}^T)^{-1}\boldsymbol{A}\boldsymbol{\Lambda}(\boldsymbol{\theta})^{-1} \tag{3.28}$$

Where $\boldsymbol{I}$ is the identity matrix. For the full derivation of $\boldsymbol{P}(\boldsymbol{\theta})$ see [12].

Using $\boldsymbol{P}(\boldsymbol{\theta})$, the hybrid force/position controller is:

$$\boldsymbol{\tau} = \boldsymbol{J}_b^T\Big(\boldsymbol{P}(\boldsymbol{\theta})\Big(\boldsymbol{\Lambda}(\boldsymbol{\theta})\Big(\dot{\mathcal{V}}_d + K_p X_e + K_i \int X_e(t)\,dt + K_d \mathcal{V}_e\Big)\Big) \tag{3.29}$$

$$\big(\boldsymbol{I} - \boldsymbol{P}(\boldsymbol{\theta})\big)\Big(\mathcal{F}_d + K_{fp}\mathcal{F}_e + K_{fi}\int \mathcal{F}_e(t)\,dt\Big) \tag{3.30}$$

$$+ \boldsymbol{\eta}(\boldsymbol{\theta})\Big) \tag{3.31}$$

This controller allows for the pushing in the x direction, while controlling the robots position in the other directions.

Since the controller works in task space, it is not necessary to calculate inverse kinematics while moving, the controller essentially calculates the inverse kinematics internally. This is both an upside and a downside to this controller. The path planning can be a lot simpler when there is only need to plan a trajectory in Cartesian space. However, in some cases it might be desirable to plan the actual joint positions instead, which is not possible using this controller.

Another downside to this controller is that there are a lot of tuning parameters ($K_p, K_d, K_i, K_{fp}, K_{fi}$). All of the tuning parameters can be implemented as scalars. However, the twists contain both position and orientation. To get desired results, it is usually necessary to have different parameters for the position and orientation. To achieve this, the tuning parameters can also be implemented as (6×1) vectors, using element-wise multiplication instead of matrix multiplication. It would also be possible to use (6×6) matrices as tuning parameters, however, that seems overly complex and should not be necessary to to achieve good results.

3.4. Adaptive control

In the previous Sections, a control scheme for the lifting of a parcel with two robot manipulators is presented. In the control scheme , the weight of the parcel is used to calculate the required force to keep friction. On the other hand, the weight of the parcel is not used to calculate the feedforward gains of the position controller. This is compensated by the large proportional feedback gain of the positional controller. In order to reduce the feedback gains and improve tracking performance, an adaptive controller scheme is used to estimate the weight of the parcel, and use it for control.

For the case of moving parcels onto a conveyor, the weight of the parcels is not known. Therefore, the first step of the adaptive controller is to estimate the weight of the parcel being lifted. This is done using the forward dynamics of the system. Than, the weight estimate is used to calculate the new mass matrix and gravity compensation for the hybrid force/position controller. Finally, trough simulation is it shown that the use of this scheme can improve tracking performance and reduce the feedback torques of the joints.

3.4.1. Weight estimation

To estimate the weight of the parcel, the forward dynamics of the robot are used. The standard equation of the forward dynamics of the robot are as follows (as given by [12]):

$$\boldsymbol{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} = \boldsymbol{\tau} - \boldsymbol{g}(\boldsymbol{\theta}) - \boldsymbol{J}_b^T\mathcal{F}_{tip} \tag{3.32}$$

In this equation, an extra wrench $\mathcal{F}_{tip}$ is introduced. This wrench represents the sum of all the forces that are acting against the robots end effector, in the body frame.

The mass of the parcel is separated from the remainder of the mass matrix M. The mass matrix is calculated as the sum of the inertia of each joint. The total mass matrix is therefore:

$$\boldsymbol{M} = \sum_{i=1}^{n} \boldsymbol{J}_{ib}^T \boldsymbol{G}_i \boldsymbol{J}_{ib}(\boldsymbol{\theta}) \tag{3.33}$$

$$\boldsymbol{G} = \begin{bmatrix} \mathcal{I} & 0 \\ 0 & m\boldsymbol{I} \end{bmatrix} \tag{3.34}$$

Here, n is the number of joints, $\boldsymbol{G}_i$ represent the spatial mass matrix of the joint i, $\mathcal{I}$ is the inertia matrix and $\boldsymbol{J}_{ib}$ is the body Jacobian matrix of the joint, where the first i columns are as in the body Jacobian, and the remainder is all 0. Since the mass matrix is a sum, we can simply split the mass matrix of the robot and the mass matrix of the parcel:

$$(\boldsymbol{M}_{robot} + m_p \boldsymbol{J}_b^T \boldsymbol{G}_p \boldsymbol{J}_b)\ddot{\boldsymbol{\theta}} = \boldsymbol{\tau} - \boldsymbol{g}(\boldsymbol{\theta}) - \boldsymbol{J}_b^T \mathcal{F}_{tip} \tag{3.35}$$

$$\boldsymbol{G}_p = \begin{bmatrix} 0 & 0 \\ 0 & \boldsymbol{I} \end{bmatrix} \tag{3.36}$$

Since we have no way of knowing the inertia matrix $\mathcal{I}$ of the parcel, and the parcel should not rotate much during lifting, the inertia is assumed to be zero. Therefore the matrix $\boldsymbol{G}_p$ of the parcel is as in equation 3.36. Note that the parcels mass m_p is also separated from this matrix.

The gravity $\boldsymbol{g}(\boldsymbol{\theta})$ also needs to be separated for robot and the parcel. The gravity is equal to partial derivative of the total potential energy with respect to $\boldsymbol{\theta}$ (see equation (3.20)). Since the potential energy is a sum of potential energy for each link, the parcel and robot can simply be separated. Separating the gravity results in the following equation:

$$m_p(\boldsymbol{J}_b^T \boldsymbol{G}_p \boldsymbol{J}_b \ddot{\boldsymbol{\theta}}) + \boldsymbol{g}_p(\boldsymbol{\theta}) = \boldsymbol{\tau} - \boldsymbol{g}_{robot}(\boldsymbol{\theta}) - \boldsymbol{J}_b^T \mathcal{F}_{tip} - \boldsymbol{M}_{robot}\ddot{\boldsymbol{\theta}} \tag{3.37}$$

Here, $\boldsymbol{g}_{robot}(\boldsymbol{\theta})$ is the gravity acting on the robot (the same as $\boldsymbol{g}(\boldsymbol{\theta})$ for the controller without weight estimation), and $\boldsymbol{g}_p(\boldsymbol{\theta})$ is the the gravity of the parcel. Since $\boldsymbol{g}_p(\boldsymbol{\theta})$ is linear for m_p, it is possible to replace it by $m_p \boldsymbol{g}_1(\boldsymbol{\theta})$, where $\boldsymbol{g}_1(\boldsymbol{\theta})$ is the gravity of the parcel for unit mass. This results in:

$$m_p\left(\boldsymbol{J}_b^T \boldsymbol{G}_p \boldsymbol{J}_b \ddot{\boldsymbol{\theta}} + \boldsymbol{g}_1(\boldsymbol{\theta})\right) = \boldsymbol{\tau} - \boldsymbol{g}_{robot}(\boldsymbol{\theta}) - \boldsymbol{J}_b^T \mathcal{F}_{tip} - \boldsymbol{M}_{robot}\ddot{\boldsymbol{\theta}} \tag{3.38}$$

$\mathcal{F}_{tip}$ is hard to know exactly. For this setup, it is assumed to be equal to the wrench in the x direction exerted by the other robot. This is not completely accurate, however, when the y and z coordinates of both robots end effectors are close, this assumption should be fairly accurate.

In this equation, the entire left side is multiplied by the parcels mass, while the parcels mass is completely absent from the right side. Therefore we can use this equation to get an estimate of the parcels weight during lifting. Both sides of the equation are vectors, representing te torque in each joint. When dividing the right side of the equation by the right side element-wise, there will be n estimates for the weight of the parcel. One estimate for each joint.

Depending on the configuration of the robot, not all of the estimates per joint might be as accurate. Therefore, a selection is made for which joints give the best estimates. The selection of the joints for the simulations in this thesis is presented in section 4.3.1.

3.4.2. Implementation

In the Section above, a scheme for estimating the weight of a parcel is presented. In this Section, the implementation of the estimated weight in the controller of Chapter 3 is presented.

Filter

In order to get more reliable and stable control, the weight estimates are not used directly. Instead, a simple filter is used, to make sure the estimated weight does not fluctuate to much. This is especially important

when the robot put the parcel back down. This can clearly be seen in Figure 4.8. At $t = 3.6$, the moment when the parcel touches the ground, the estimate becomes very unreliable.

The formula for the weight estimation filter is as follows:

$$\hat{m}(k) = \hat{m}(k-1) + K_{filter}\Big(\frac{\sum_{i=1}^{n_e} \hat{m}_i(k)}{n_e} - \hat{m}(k-1)\Big) \tag{3.39}$$

Here $\hat{m}_i(k)$ is the estimate for m for the ith joint that is used for estimation at the current time step, n_e is the number of joints used for estimation and $\hat{m}(k)$ is the total estimate of m at the current time step. Finally, K_{filter} is the filter gain.

Since there are 2 robot, each carrying about half the weight of the parcel, the total weight of the parcel is simply the sum of the 2 weight estimates. For control it might however seem better to keep these two weight estimates separate, for the case that the parcels center of mass in not in the center of the parcel. In that case, one of the robots will carry more weight than the other robot. By keeping the weight estimates separate, the weight estimate for each robot more accurately represent the actual weight the robot has to carry. In practice however, it seems to be quite difficult to get a stable system using this method. This is because when the estimated weight for one of the robots goes down, it uses less force to lift, which the other robot has to compensate. That way the estimated mass for the second robot keeps increasing while the weight goes down for the other robot. This usually results in one of the robot losing contact with the parcel, and the system failing.

Control
Now that there is estimation for the weight of the parcel being manipulated, it can be implemented in the controller. To implement the full dynamic model of the parcel is quite complicated, and is also unnecessary, since the parcel does not rotate during the standard picking and placing operation. Instead, the parcel is simply modeled as a point mass a the tip of the end effector of the robot. This approach is justified, because the the parcel that is being lifted at two contact points has its weight distributed at these contact points. (This approximation is very similar to the approximation that is already used for the weight estimation.) For the gravity compensation, the parcel is also modeled as a point mass at the tip of the end effector.

The new mass matrix of the robot if formulated as:

$$\hat{\boldsymbol{M}}(\boldsymbol{\theta}) = \boldsymbol{M}_{robot}(\boldsymbol{\theta}) + \boldsymbol{J}_b^T \boldsymbol{G}_p \boldsymbol{J}_b \hat{m} \tag{3.40}$$

Here $\boldsymbol{G}_p$ is once again the parcels spatial dynamics for unit mass.

The gravity is formulated as:

$$\hat{\boldsymbol{g}}(\boldsymbol{\theta}) = \frac{\partial P_{robot}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \frac{\partial \hat{P}_p(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{3.41}$$

Here $P_p(\boldsymbol{\theta})$ is the position of the parcel in the z direction multiplied by $\hat{m}$.

Since the weight of the parcel is initially unknown, but it is required do find the required internal force, it needs to be set to some weight. For the experiments, this mass is set to the maximum weight this system is able to lift. This does mean that for lighter parcel the weight is a lot higher than necessary. In a real system finding the actual required force to keep the parcel from slipping is a lot harder, since it also depends on the friction coefficient of the parcels material. The desired external force in the x direction is calculated using $\hat{m}$.

3.5. Conclusion

In this Chapter, a control scheme is presented for lifting a parcel with two robotic manipulators, using only friction to keep the parcel from moving.

In order to improve tracking properties of the system, and reduce the required feedback torque, an adaptive controller is also presented in this chapter. The adaptive controller estimates the weight of the parcel as

the robots are lifting it. The estimated weight is implemented in the model of the robot dynamics, to improve the accuracy of the feedforward torque, and to reduce the amount of feedback torque required, as well as improving the tracking properties of the system.

The effectiveness of the controller with and without the adaptive part is demonstrated in Chapter 4. It is validated that the controller can achieve desirable results for tracking a trajectory with a parcel. It is than shown that the adaptive controller can find good estimates for the weight of a parcel, and improve the tracking properties of the system.

4

Simulation results

In this Chapter, the proposed controller will be tested for robustness. Since the actual setup has to move a lot of parcels, the system has to move quite quickly. This makes it a lot more difficult to have a low failure rate. In this chapter, the balance between reliability and speed will be investigated, by testing the system for an assortment of parcel, with different shapes, masses and centers of gravity.

4.1. PyBullet

The simulation preformed in this Chapter are in PyBullet [5]. PyBullet is a python extension for the bullet physic library. In this program, dynamics of robots and other objects can be simulated. All the code to set up the simulation, as well as the controllers are run in Python.

The simulation consists of two KUKA iiwa robots [5]. The KUKA iiwa robot is a small industrial robot manufactured by KUKA GmbH. The robot is a redundant manipulator, which means that it has more joints than there are Cartesian coordinates, so there are 7 joints while there are only 6 Cartesian coordinates in 3D space. This means that the robot can move around while the end effector stays in the exact same position, something that 6 joint robot manipulators can not use. For the hybrid force position controller however, there needs to be a one to one mapping from joint space to Cartesian space (the Jacobian should be invertible). Therefore the third joint of the iiwa robot is not used, but fixed at its 0 position. That way the robot reduces to a standard 6R robot, with a spherical wrist. The spherical wrist means that the end effector of the robot can rotate in all directions using only the last 3 joints and without changing position of the wrist.

The robots have a sphere attached to their end-effectors, which will be called the fingertips of the robot. This sphere will have a more consistent contact surface with the parcels. It also makes it easier to change the friction coefficient of the robots fingertips. Further, there is a box loaded in in the simulation, which represents a parcel. All of the files are loaded in the simulation as URDF files. Using the XML-tree extension, we can easily edit these URDF files with Python, to change the shape and dynamics of the parcel.

There are one other controller implemented in the Python code, which is a standard joint space controller, that moves the robot from one configuration in joint space to the next one. This controller will be used to move the robot when the simulation starts, as the robot starts completely upright. This position is a singularity, which means that the Jacobian matrix no longer has full rank. This makes it impossible for the robot to move in some direction in Cartesian space. Since the hybrid force position controller requires the inverse of the Jacobian matrix (there needs to be a one to one mapping from Cartesian coordinates to joint coordinates) which does not exist if the robot is at a singularity, the joint space controller will move the robot to a new position, which is not at a singularity. From there, the hybrid force position controller will control the position of the robot. To move to the correct position using the joint space controller, there will also be an inverse kinematics calculator. The inverse kinematics calculator takes a position in Cartesian space, and calculates the position in joint space that moves the end-effector of the robot to that position. The inverse kinematics starts with a simple initial guess and than optimizes the position numerically.

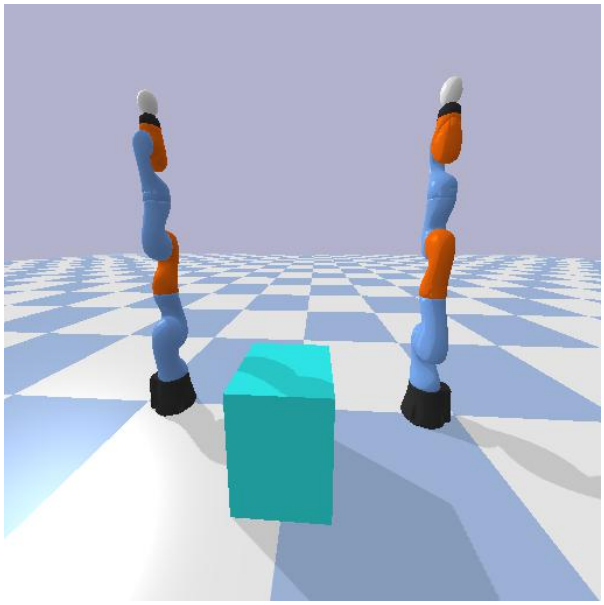


Figure 4.1: An image of the simulation

4.2. Simulation results without adaptive control

In this section, the controller from Chapter 3 is tested, and the results are discussed.

The hybrid force position controller and the force controller are used for lifting a parcel. In the simulation, the robots start at the zero position. That means that the robot is pointing straight up. Since this position is a singularity, the task space position controller cannot be used directly, since the controller requires the inverse of the Jacobian. Therefore, a joint position controller is used, to move the the robot to the starting position. From there, the hybrid force position controller is used to move the robots to the parcel. During this motion, no force needs to be used in the x direction. Therefore the constraint matrix A is equal to a (1×6) matrix with all zeros.

Once in position, the control scheme above is used to grab and lift the parcel, and follow the desired trajectory. The normal force at the contact points is used as force feedback. The positional feedback is equal to the position calculated by the joint positions using the forward kinematics of the robot. The force controller uses the actual parcel position directly, however, this could easily be changed to also use the robots joint positions.

4.2.1. Controller parameters

For the simulation, a KUKA iiwa robot is used [1]. As mentioned in Chapter 3, the robot needs to have six degrees of freedom, in order for the Jacobian to be invertible. Because the KUKA iiwa has 7 degrees of freedom, the thirds joint is unused, and locked at the zero position. This reduces the KUKA iiwa to a more standard 6R robot.

For the force controller, the following parameters are used:

$$K_p = 500$$
$$K_d = 10$$

And the control parameters for the hybrid force position controller are:

$$K_p = \begin{bmatrix} 10000 \\ 10000 \\ 10000 \\ 1000 \\ 1000 \\ 1000 \end{bmatrix} \quad K_i = \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{bmatrix} \quad K_d = \begin{bmatrix} 20 \\ 20 \\ 20 \\ 20 \\ 20 \\ 20 \end{bmatrix}$$

$$K_{fp} = 5 \qquad K_{fi} = 1$$

As mentioned before, the proportional gain for the position is quite large. This is because the weight of the parcel is not known, and the controller needs to compensate for the unknown mass. With a lower proportional gain, the robot would not track the position properly when lifting heavy parcels.

4.2.2. Trajectory tracking

We would like to see if the controller can adequately track a trajectory for the parcel. Therefore a couple of trajectories are constructed. The results of these simulation is plotted.

The first trajectory is a short trajectory in which the robots lift the parcel at $x = 0, y = -0.4, z = 0$ and place it at $x = 0, y = 0.4, z = 0$. The robots start pushing on the parcel at $t = 0$s, start lifting at $t = 1$s, and places the parcel down at $t = 5$s. The parcels mass for this simulation is $m = 5$kg. The results are plotted in Figure 4.2. From this plot we can see that the controller tracks a trajectory relatively well. However, in the z direction we can see that the actual position is always a few centimetres lower than the desired position during lifting. This is because the parcels weight is not implemented in the mass matrix and the gravity compensation. Secondly, since the positions in y and z directions for feedback are retrieved from the forward kinematics of the robot, any slipping of the parcel will also cause an offset for the remainder of the trajectory.

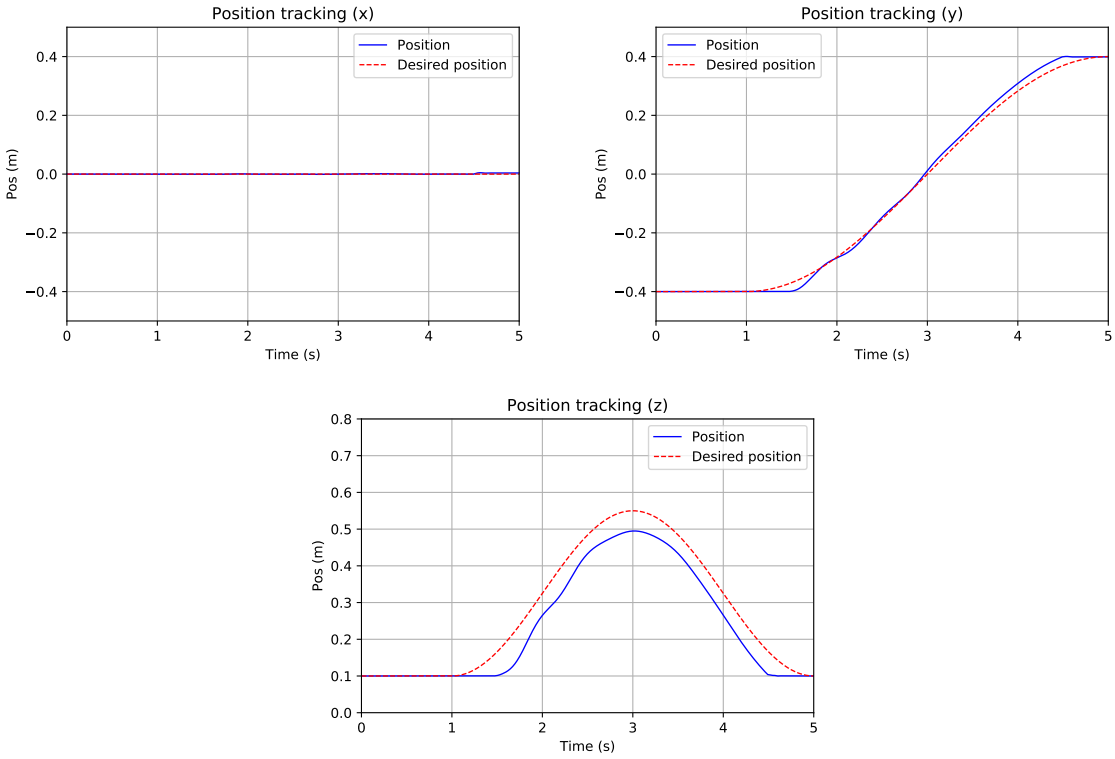


Figure 4.2: Trajectory tracking with a parcel of mass $m = 5$ kg

We would also like to see if the controller is able to track trajectories in the x direction. Therefore a second trajectory is constructed, in which the robots lift the parcel, and than track a sinus wave in the x direction.

The results for the trajectory planning for this trajectory are plotted in Figure 4.3. We can clearly see that the robot properly tracks the trajectory in the x direction, while also tracking the other directions fairly well.

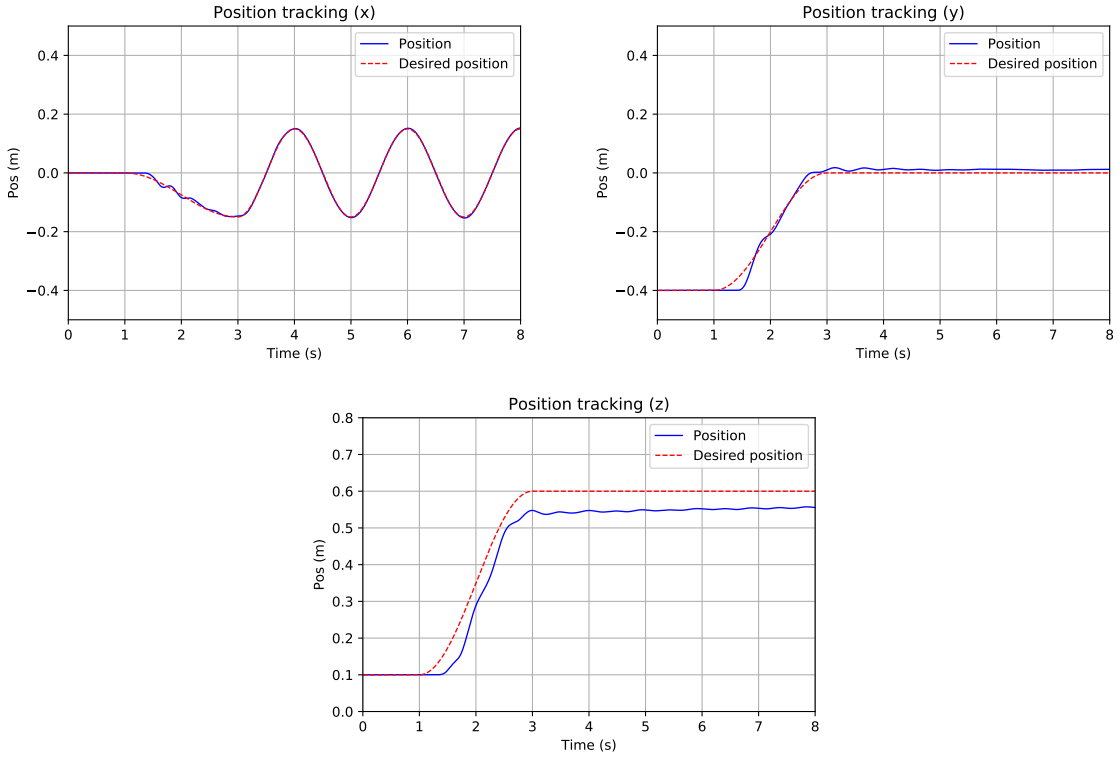


Figure 4.3: Trajectory tracking with a parcel of mass $m = 5$ kg

4.2.3. Forces

Proper tracking of the desired forces is investigated next. It is important that the desired forces as calculated by the force controller are properly tracked by the hybrid force position controller, since improper tracking would cause slipping if the forces are to low at any point. On the other hand, if the forces get to large, it could damage the parcels. The contact force and desired force for one of the manipulators during the lifting operation from figure 4.3 are plotted in 4.4.
From the image it can be seen that forces are tracked fairly well. There is however a small offset during the lifting. This can easily be overcome by increasing the integral gain of the force part, of the hybrid force position controller. By increasing the integral gain to $K_{fi} = 20$ (up from 1), the tracking of the desired force is a lot more accurate, see figure 4.5.
Both plots show a large peak at the start of the graph. This is the moment the robots first make contact with the parcel. The peak is caused by the dynamic forces which build up in the robot before finding contact with the parcel. Building up the forces more slowly might make the peak a smaller, however, this would also slow down the system a bit, and since the forces are not critically high, the controller is left as it is. Some small peaks also show at the end of the trajectory, which is the moment the parcel again makes contact with the floor.

The plots show that, when using the correct tuning parameters, accurate force tracking can be achieved. This allows for choosing a small safety margin in Equation (3.4). When the force tracking is accurate it will also significantly decrease the chance of any parcel being damages by the manipulators.

4.2.4. Feedforward & feedback

To get a better idea of how the controller preforms, we take a look at the ratio between the feedforward and the feedback components of the controller during a simple trajectory. The trajectory is a short trajectory in

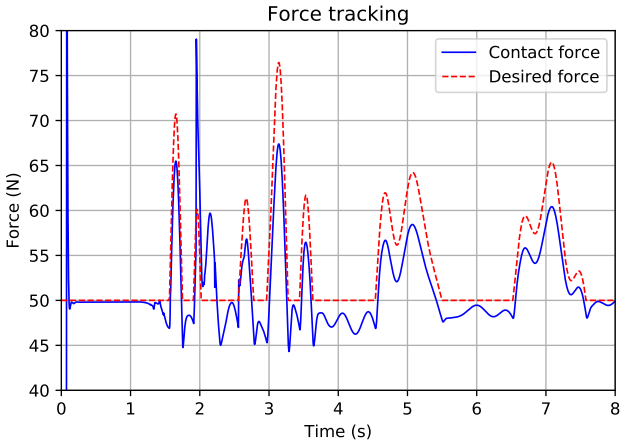


Figure 4.4: Force tracking of the hybrid force position controller

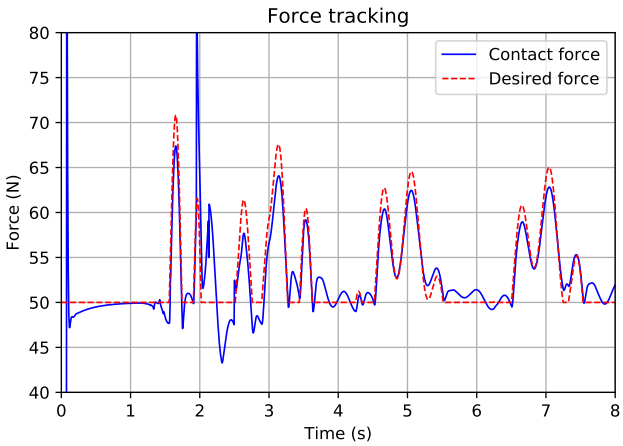


Figure 4.5: Force tracking of the hybrid force position controller

which the robots lift the parcel at $x = 0, y = -0.4, z = 0$ and place it at $x = 0, y = 0.4, z = 0$. The robots start pushing on the parcel at $t = 0$, start lifting at $t = 1$, and places the parcel down at $t = 5$. The parcels mass for this simulation is $m = 1$kg. The following graphs show the feedforward and feedback torques on each joint. Since the torques are a linear combination of the feedback and feedforward wrenches in task space, we can split them. The feedforward en feedback torques are calculated as:

$$\boldsymbol{\tau}_{ff} = \boldsymbol{J}_b\Big(\boldsymbol{P}\boldsymbol{\Lambda}\dot{\mathcal{V}}_d + (\boldsymbol{I}-\boldsymbol{P})\mathcal{F}_d\Big) + \boldsymbol{J}_b\boldsymbol{\eta}$$
$$\boldsymbol{\tau}_{fb} = \boldsymbol{J}_b\Big(\boldsymbol{P}\boldsymbol{\Lambda}\Big(K_p X_e + K_i\int X_e + K_d\mathcal{V}_e\Big) + (\boldsymbol{I}-\boldsymbol{P})\Big(K_{fp}\mathcal{F}_e + K_{fi}\int \mathcal{F}_e(t)dt\Big)\Big)$$

The joint feedforward torques and feedback torques are plotted in Figure 4.6. From these graphs we can see that the overall torques in the first three joints are much larger than the last three. This is to be expected, since the first joints have to move a lot more mass of the robot itself. We can see that for these first three joints the feedforward forces are generally a lot larger than the feedback forces. Ideally we would want the the feedback forces to remain close to zero, as this would indicate the model is accurate. We can also see two moments with very high peaks in the control inputs. The first one is at $t = 0.2$. This is the moment the robots first make contact with the parcel, causing a large deceleration in some of the joints, which causes a high control action. At this moment, the contact force also gets very large momentarily, which causes the force controller of the hybrid controller to compensate. The rapid change in torque could cause problem for te robots actuators. They are easily overcome however, by lowering the tuning parameters of the hybrid force position controller, until the robot makes contact with the parcel. For the remainder of the simulation these peaks do not impact the result, so therefore the controller is left as it is. The second moment the controller peak is at $t = 4.6$, which is the moment the parcel again makes contact with the floor. This also causes some rapid changes in the velocity of the end-effector, and the contact forces.

We will now increase the weight of the parcel to $m = 5$kg (see Figure 4.7). Now we can clearly see that the feedback components, especially for the second and third joint, have increase a lot. For joint 5, we can also see that the feedback torque is significantly higher than the feedforward torque. This is not desirable, and indicates that the model of the robot does not preform very well. This of course has to do with the fact that the mass of the parcel is not incorporated in mass matrix or gravity model. In the next Section, it is shown that the adaptive controller can overcome this issue.

4.3. Simulation results for adaptive control

The adaptive controller will be tested and compared to the controller without the weight estimation to see if it improves the tracking property of the controller, and if it reduces the required feedback.

4.3.1. Implementation

For the implementation of the estimator, a simple filter is used. However, since the estimator gives one estimate per joint, it should first be decided which joints give the best results. We will decide what joints to use for estimation by plotting their individual estimates during a short lifting operation, as well as looking at their mean squared error.

Estimates per Joint

First we take a look at the mean and mean squared error of the estimate of the parcels weight given by each joint of one of the two robots. For this operation, the parcels weight is 4 kg. Since the robot each lift half of the parcel, the weight the robot should estimate is 2 kg. The mean and mean squared error for each joint are shown in Table 4.1. The weight estimates per joint during the trajectory for joints 2 to 5 are also plotted in figure 4.8.

joint	1	2	3	4	5	6
mean $\hat{m}$ (kg)	-1.003	1.998	2.031	2.964	2.633	-2.853e12
mean squared error $\hat{m}$ (kg)	4740.5	0.0151	0.0024	23.654	0.4035	2.89e27

Table 4.1: Mean and mean squared error for $\hat{m}$ for each joint

We can see that for joint 1 and 6, the estimate if very far off. For joint 1 this is because it is parallel to the direction of the gravity. Therefore the gravity does not effect its dynamics, making it a lot more difficult to

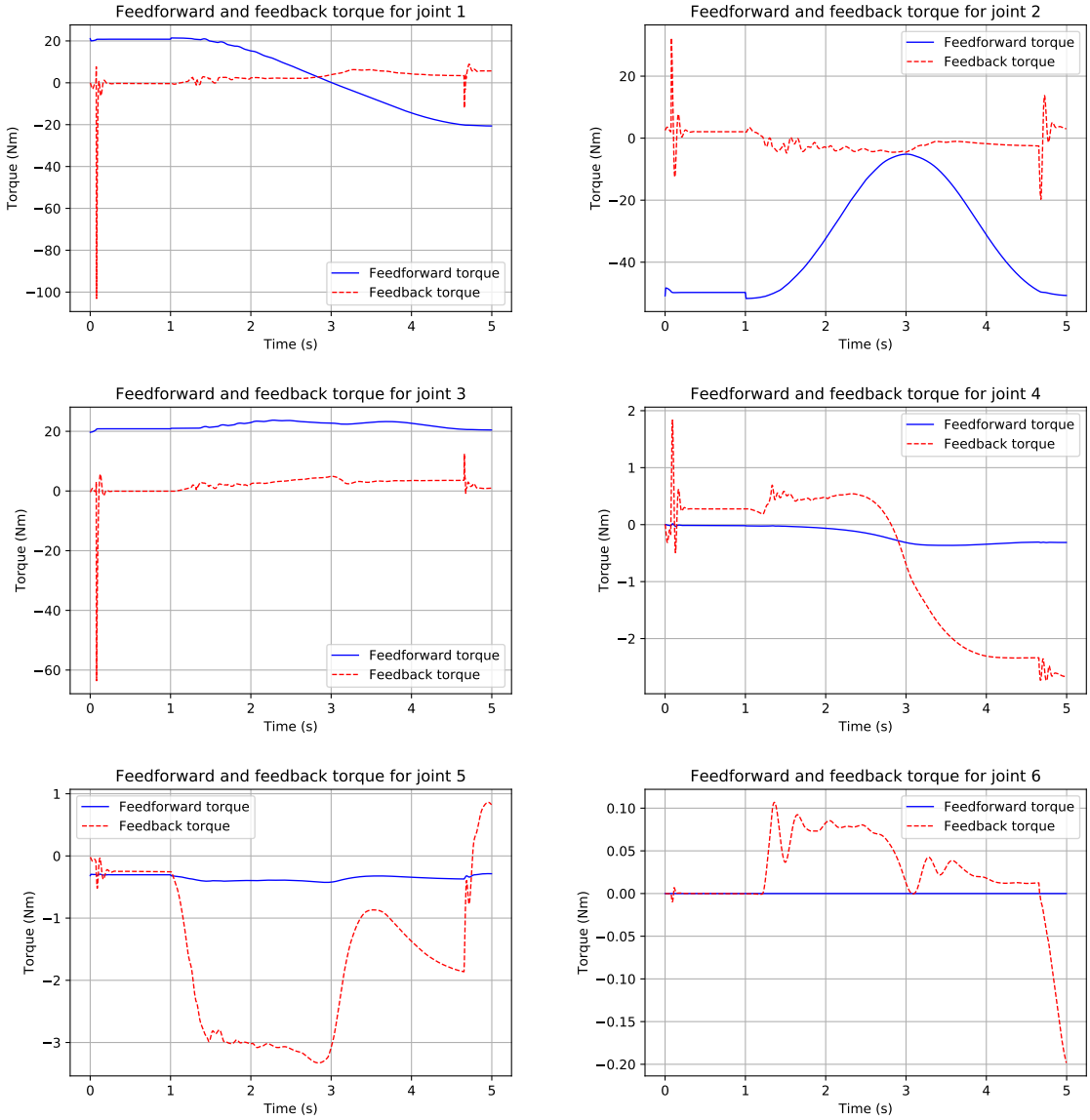


Figure 4.6: Feedforward and feedback components of joint torques with parcel mass of 1 kg

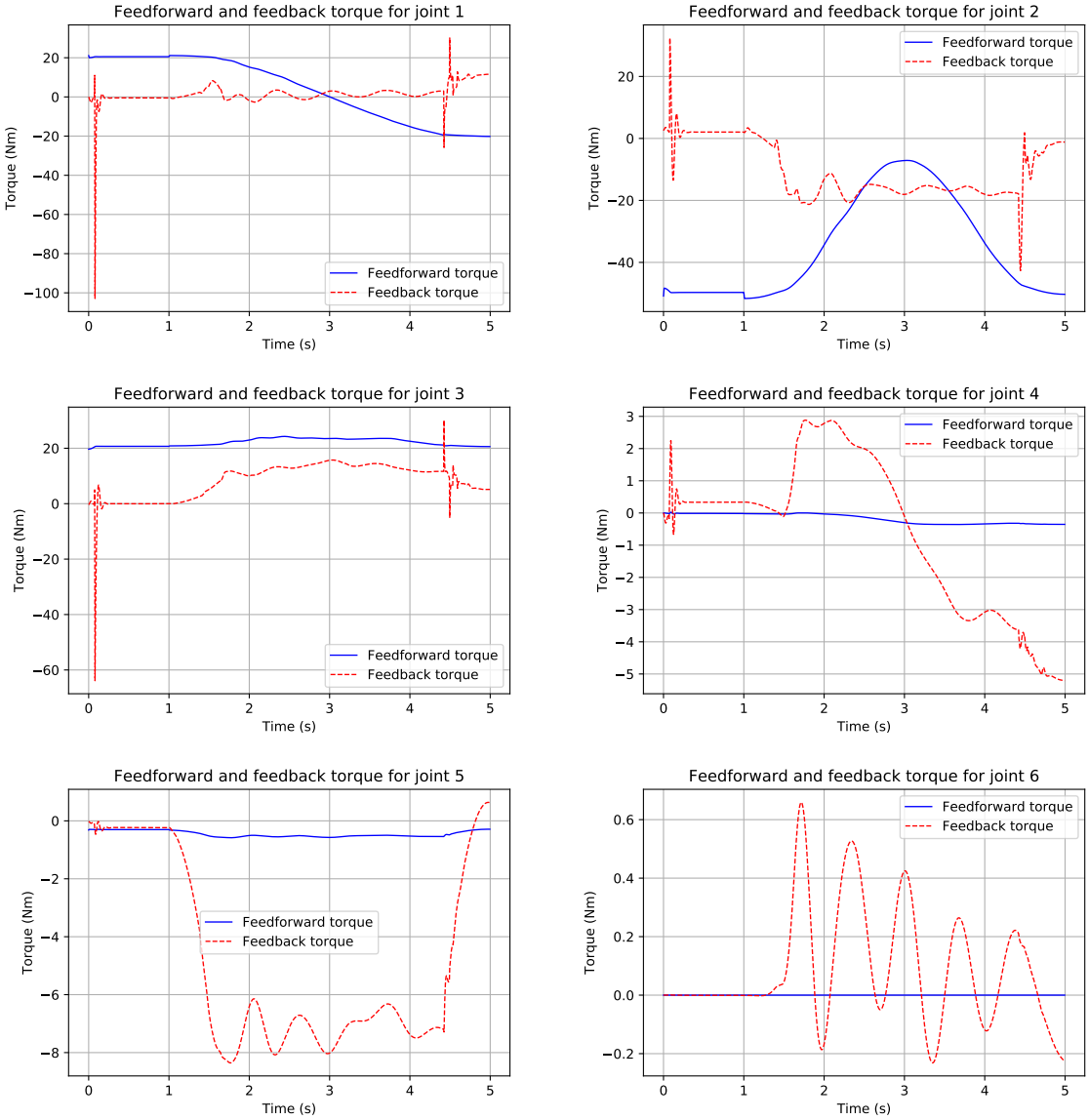


Figure 4.7: Feedforward and feedback components of joint torques with parcel mass of 5 kg

estimate the parcels weight. Joint 6 is in a singularity regarding the parcels weight (since the inertia of the parcel is assumed to be 0). Therefore it is not possible to estimate the parcels weight using joint 6.

More interesting is joint 4. The estimate for joint 4 is fairly accurate during most of the lifting, however, when the robots move trough the y plane, the estimate goes up to a very high number, and than drops down to a very low number (see figure 4.8). This indicates that the estimate for the mass as seen from joint 4 moves trough a singularity at that point, making the mass not observable for a short time. This causes the higher means square error of joint 4. Joint 5 does give us a more stable estimate for the weight of the parcel, however, the mean of this estimate is still quite a bit higher than the actual value.

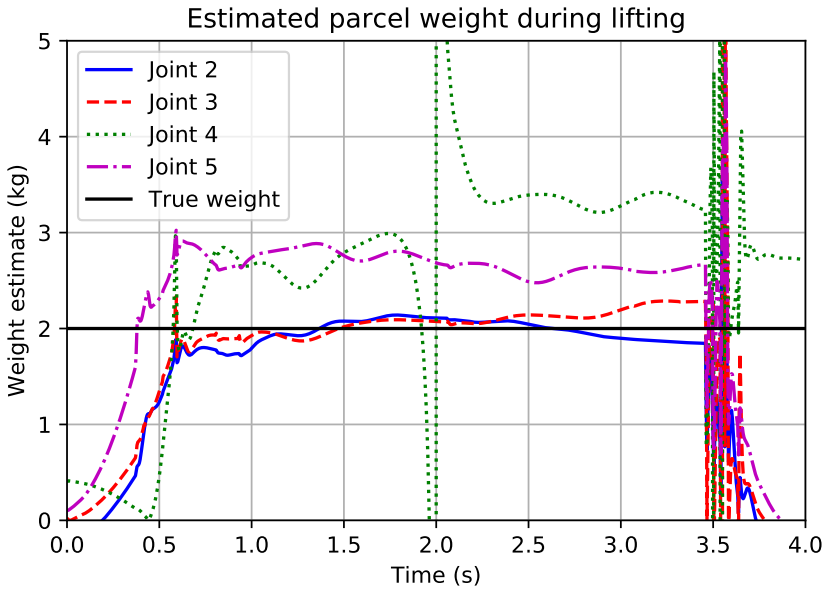


Figure 4.8: Weight estimate of the parcel during lifting per joint

Clearly joint 2 and 3 give us the best estimate for the mass of the parcel. Therefore, a filter will be designed using these two estimates.

Filter

In order to get more reliable and stable control, the weight estimates are not used directly. Instead, a simple filter is used, to make sure the estimated weight does not fluctuate to much. This is especially important when the robot put the parcel back down. This can clearly be seen in figure 4.8. At $t = 3.6$, the moment when the parcel touches the ground, the estimate becomes very unreliable.

The formula for the weight estimation filter is as follows:

$$\hat{m}(k) = \hat{m}(k-1) + g\left(\frac{\hat{m}_2(k) + \hat{m}_3(k)}{2} - \hat{m}(k-1)\right) \tag{4.1}$$

Here $\hat{m}_i(k)$ is the estimate for m for joint i at the current time step, and $\hat{m}(k)$ is the total estimate of m at the current time step. g is the filter gain.

Since there are 2 robot, each carrying about half the weight of the parcel, the total weight of the parcel is simply the sum of the 2 weight estimates. For control it might however seem better to keep these two weight estimates separate, for the case that the parcels center of mass in not in the center of the parcel. In that case, one of the robots will carry more weight than the other robot. By keeping the weight estimates separate, the weight estimate for each robot more accurately represent the actual weight the robot has to carry. In practice however, it seems to be quite difficult to get a stable system using this method. This is because when the

estimated weight for one of the robots goes down, it uses less force to lift, which the other robot has to compensate. That way the estimated mass for the second robot keeps increasing while the weight goes down for the other robot. This usually results in one of the robot losing contact with the parcel, and the system failing.

The estimated weight of the parcel during a simple pick and place task is plotted in figure 4.9. The first plot is for a 4 seconds trajectory, while the second plot follows the same path, but with a 2 second trajectory. A filter gain of $K_{filter} = 0.05$ is implemented, for a refresh rate of 240 Hz. From the figure it can be seen that as soon as the robot lifts the parcel fully, around $t = 0.5$ seconds for the slower trajectory, the estimate of the weight is fairly accurate. For the faster trajectory, the weight does not get as accurate. There is also a small oscillation that can be seen at the start of simulation. This is causes by the parcel swinging slightly between the two contact points. This is caused by the fast movement of the robots. This can be reduced by increasing the contact surface of the robot manipulators, increasing the friction.

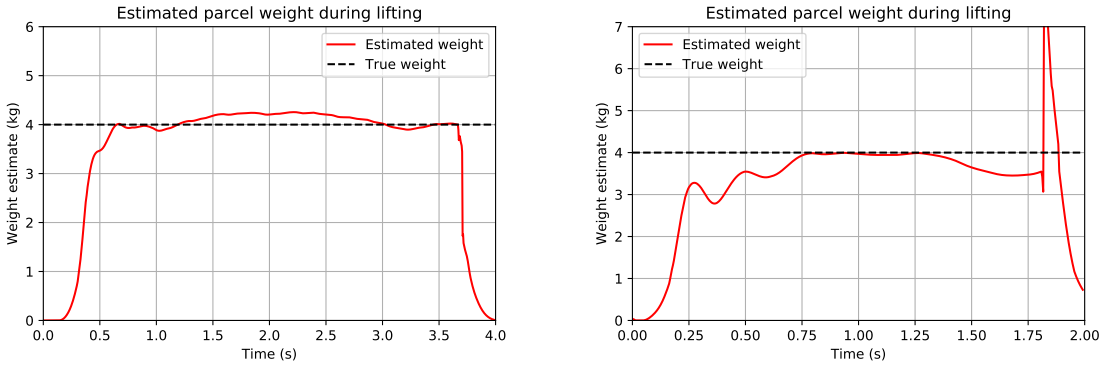


Figure 4.9: $\hat{m}$ during a 4 and 2 second pick and place trajectory with $m = 4$ kg

4.3.2. Trajectory tracking

A picking and placing trajectory is plotted in Figure 4.10. The trajectory and parcels weight is similar the the trajectory in the previous chapter. We can clearly see that the adaptive controller has a significant improvement for the trajectory tracking property, especially in the z direction. This is because the gravity compensation now has a lot better information about the actual weight is has to compensate for. The tracking property is also evaluated by calculating the means square error of the trajectory (see Table 4.2). The mean squared error has been reduced significantly, indicating the improved trajectory tracking property of the adaptive controller.

	Mean square error
Adaptive controller	0.00046
Non-adaptive controller	0.00305

Table 4.2: Mean squared error of the position during the trajectory of Figure 4.10 (Adaptive) and Figure 4.2 (non-adaptive)

4.3.3. Feedforward & feedback

The behaviour of the feedforward and feedback dynamics of the adaptive controller is also investigated. Reducing the required amplitudes of the feedback signals is important, since it will make the system more stable and reliable. For the same picking and placing trajectory, the feedforward and feedback torques per joint are plotted in Figure 4.11. When compared to the feedforward and feedback torques for the same trajectory without the adaptive controller, as seen in Figure 4.7, it is clear the the feedback has been reduced significantly. The reduction of the required feedback signals makes for a better and more stable performance of the controller. It indicates the usefulness and effectiveness of the proposed adaptive controller.

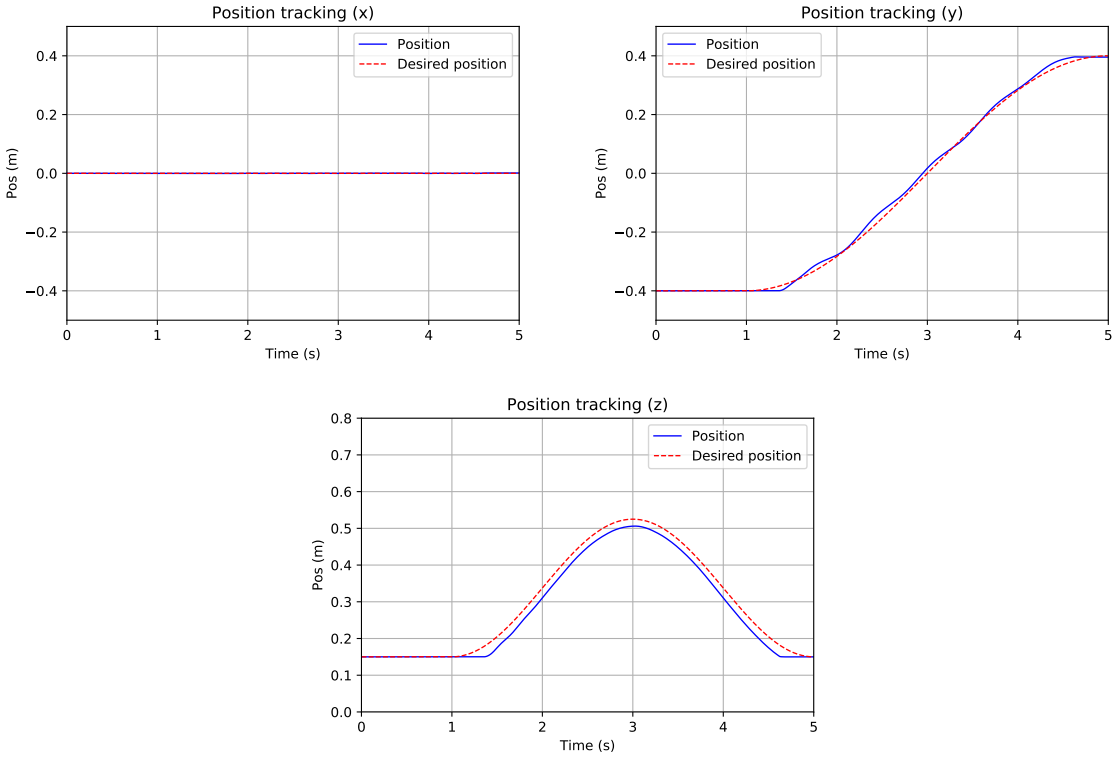


Figure 4.10: Trajectory tracking for a 5 second trajectory with parcel mass of 5 kg

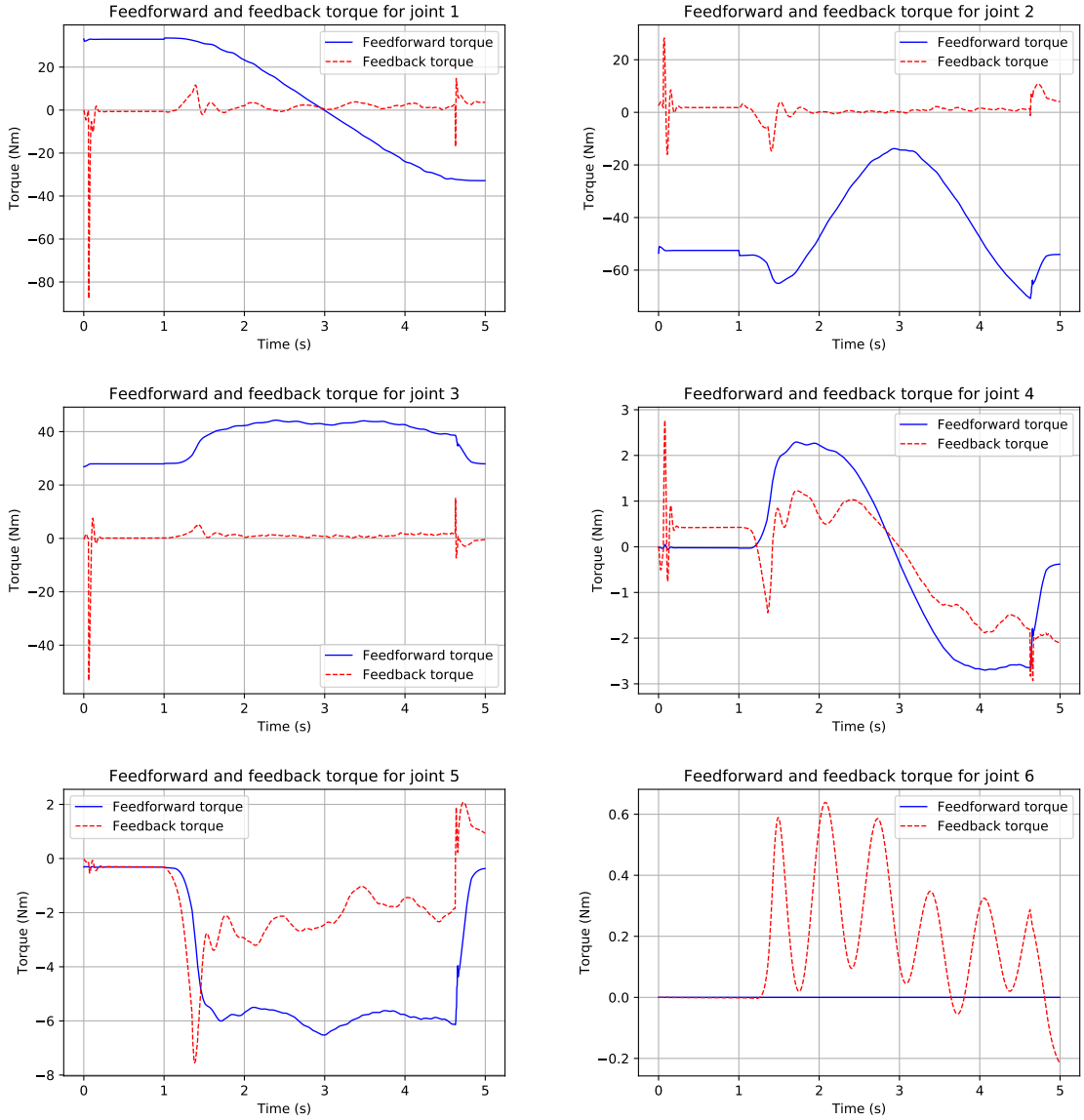


Figure 4.11: Feedforward and feedback components of joint torques during a pick and place trajectory with parcel mass of 5 kg

4.4. Robustness

To test for the robustness of the system, an assortment of parcels will be moved. At the end of the trajectory, the parcels position is measured, to see if the parcel made it to the end, or if the system failed.

The system is tested for a 2 second picking and placing trajectory. The size of the parcel, mass of the parcel, center of mass are varied. Some trajectories will also have a different starting position for the parcel, to see if the system also preforms well when the parcel leaves this $x = 0$ plane. This makes a big difference, as the two robots are no longer symmetric with each other when this happens. Unfortunately, the starting position of the parcel can not be shifted too much, as the robots reach the end of their workspace.

For the first test, some parcels with different mass and sizes are lifted along a 2 second trajectory. For now, the center of gravity is in the middle of the parcel, and starting position is at $x = 0$. The results of this test are shown in Table 4.3.

mass	Size (x,y,z)	Center of mass (x,y,z)	x	t	Success
1	(0.2 0.2 0.2)	(0 0 0)	0	2	True
3	(0.2 0.2 0.2)	(0 0 0)	0	2	True
5	(0.2 0.2 0.2)	(0 0 0)	0	2	True
6	(0.2 0.2 0.2)	(0 0 0)	0	2	True
7	(0.2 0.2 0.2)	(0 0 0)	0	2	True
8	(0.2 0.2 0.2)	(0 0 0)	0	2	False
1	(0.3 0.3 0.3)	(0 0 0)	0	2	True
3	(0.3 0.3 0.3)	(0 0 0)	0	2	True
5	(0.3 0.3 0.3)	(0 0 0)	0	2	True
6	(0.3 0.3 0.3)	(0 0 0)	0	2	True
7	(0.3 0.3 0.3)	(0 0 0)	0	2	True
8	(0.3 0.3 0.3)	(0 0 0)	0	2	True
1	(0.4 0.4 0.4)	(0 0 0)	0	2	True
3	(0.4 0.4 0.4)	(0 0 0)	0	2	True
5	(0.4 0.4 0.4)	(0 0 0)	0	2	True
6	(0.4 0.4 0.4)	(0 0 0)	0	2	True
7	(0.4 0.4 0.4)	(0 0 0)	0	2	True
8	(0.4 0.4 0.4)	(0 0 0)	0	2	True

Table 4.3: Table with results for parcel with different sizes and weights

From the table, it can be seen that the system fails for a parcel with a mass of 8 kg. This is near the limit of what weight the system can handle.

For the next test, the center of gravity is varied for parcel with the same size, to see how the system deals with a center of mass that is not directly in the middle of the parcel. The weight is also increased over some tests. The results of this test are shown in table 4.4. From this table we can see that moving the center of mass does make it harder for the system to correctly move the parcel, even for lower weights. This is mostly because the parcel slightly rotates, causing the robot to slip. When this happens, the adaptive controller actually hurts the system a bit, because now the estimated mass for the last link is to high, which causes the arm to oscillate. For all of these parcels, the center of mass is displaced by $\frac{1}{4}$ of the parcel dimensions, which is quite extreme. For actual parcel, it is not very likely that they are both heavy, and have a center of mass that lies that far from the center of the parcel.

The ability of the system to move outside of the x plane is also investigated. For these test, the initial position of the parcel is not directly in between the two robots. Since the robots range is limited, both in moving parcel to far away and to close, the parcel can not be moved very far. The robot can only reach about 10 cm past the $x = 0$ plane (at $y = -0.4$). The parcels are moved to the same final position, at $x = 0$ and $y = 0.4$. The results of this test are shown in Table 4.5.
Clearly, the system can still produce decent results, even when the starting position is slightly off center.

mass	Size (x,y,z)	Center of mass (x,y,z)	x	t	Success
5	(0.3 0.3 0.3)	(0.075 0 0)	0	2	False
5	(0.3 0.3 0.3)	(-0.075 0 0)	0	2	True
5	(0.3 0.3 0.3)	(0 0.075 0)	0	2	True
5	(0.3 0.3 0.3)	(0 -0.075 0)	0	2	True
5	(0.3 0.3 0.3)	(0 0 0.075)	0	2	True
5	(0.3 0.3 0.3)	(0 0 -0.075)	0	2	True
6	(0.3 0.3 0.3)	(0.075 0 0)	0	2	True
6	(0.3 0.3 0.3)	(-0.075 0 0)	0	2	False
6	(0.3 0.3 0.3)	(0 0.075 0)	0	2	True
6	(0.3 0.3 0.3)	(0 -0.075 0)	0	2	True
6	(0.3 0.3 0.3)	(0 0 0.075)	0	2	True
6	(0.3 0.3 0.3)	(0 0 -0.075)	0	2	True
7	(0.3 0.3 0.3)	(0.075 0 0)	0	2	False
7	(0.3 0.3 0.3)	(-0.075 0 0)	0	2	False
7	(0.3 0.3 0.3)	(0 0.075 0)	0	2	True
7	(0.3 0.3 0.3)	(0 -0.075 0)	0	2	False
7	(0.3 0.3 0.3)	(0 0 0.075)	0	2	False
7	(0.3 0.3 0.3)	(0 0 -0.075)	0	2	True

Table 4.4: Table with results for parcel with positions of the center of mass

mass	Size (x,y,z)	Center of mass (x,y,z)	x	t	Success
5	(0.2 0.2 0.2)	(0 0 0)	0.1	2	True
5	(0.3 0.3 0.3)	(0 0 0)	0.1	2	True
5	(0.2 0.2 0.2)	(0 0 0)	0.2	2	True
5	(0.3 0.3 0.3)	(0 0 0)	0.2	2	True
6	(0.2 0.2 0.2)	(0 0 0)	0.1	2	False
6	(0.3 0.3 0.3)	(0 0 0)	0.1	2	True
6	(0.2 0.2 0.2)	(0 0 0)	0.2	2	True
6	(0.3 0.3 0.3)	(0 0 0)	0.2	2	True

Table 4.5: Results for parcel with positions of the center of mass

For the final tests, the speed of the trajectory is decreased. The trajectory time is increased to 4 seconds. Hopefully, decreasing the speed of the trajectory will increase the stability of the system. Therefore, some of the parcels that failed with a 2 second trajectory are tested with a 4 second trajectory.

mass	Size (x,y,z)	Center of mass (x,y,z)	x	t	Success
8	(0.2 0.2 0.2)	(0 0 0)	0	4	False
7	(0.3 0.3 0.3)	(0.075 0 0)	0	4	True
7	(0.3 0.3 0.3)	(0.075 0 0)	0	4	True

Table 4.6: Results for some parcels that failed when using a faster trajectory, for a slower 4 second trajectory

As we can see in Table 4.6, slowing down the trajectory will make the system more robust in some cases, like for a center of gravity that is move very far to some side of the parcel. However, it will still have similar problems with other cases, especially for heavier parcels.

From the robustness tests, we can see that the robot does not fail very often for standard cases. Only extreme cases, where the center of gravity is shifted a lot, or the mass of the parcel is very large, can cause some trouble for the system. For lighter parcels, the system very rarely fails.

4.5. Joint limits

The controller should always stay within the limits of the joint of the manipulator, such as the velocity and torque limits. In this section, this is investigated, to see if the controller stays within the joint limit, or if some

changes need to be made to the system in order for the controller to act within these limits. Even though these limits are a bit arbitrary, since the depend on the robot that is in use, and not so much on the controller, it is still important to keep robot limitations in mind. Especially since the trajectory does not consider these limits, and only gives a trajectory in Cartesian space. These Cartesian coordinates are converted to joint space by the controller itself by way of the Jacobian.

4.5.1. Velocity limits

For this test, a simple trajectory of 2 seconds is tracked, with a parcel that has a mass of 3 kg. The maximum joint velocity for each joint, and the time during the trajectory that the maximum velocity occurred. These velocities are compared to the maximum velocities given by the technical data sheet of the KUKA iiwa [5] (note that the third joint of the robot is not used in this simulation, so joint 4 becomes joint 3 etc.). The results are shown in table 4.7

Joint	1	2	3	4	5	6
Maximum velocity [rad/s]	1.47	1.61	1.16	3.62	1.48	3.03
Allowed maximum velocity [rad/s] [1]	1.48	1.48	1.31	2.27	2.35	2.35
t [s] at max velocity	0.93	0.49	0.23	1.0	0.72	1.01

Table 4.7: Maximum velocity of each joint for a 2 second trajectory

Clearly, the maximum velocity of some of the joints is higher than the allowed maximum velocity. Especially joint 4 and joint 6 go over the limit. This occurs around $t = 1$, which is the moment the parcel crosses the $y = 0$ plane. At that moment, the wrist of the robot has to turn quickly in order to progress with the trajectory. The most straightforward solution to the problem would be to slow down the trajectory, however, since the trajectory is very simple, some optimization might also be able to solve this problem, without reducing the overall speed of the trajectory too much.

The maximum velocities per joint for a trajectory of 3 and 4 seconds are plotted in tables 4.8 & 4.9. By choosing a slower trajectory we can indeed make sure the joint velocities stay within their limit.

Joint	1	2	3	4	5	6
Maximum velocity [rad/s]	0.98	1.04	0.86	2.40	0.99	1.98
Allowed maximum velocity [rad/s] [1]	1.48	1.48	1.31	2.27	2.35	2.35
t [s] at max velocity	1.54	0.77	0.31	1.50	1.92	1.48

Table 4.8: Maximum velocity of each joint for a 3 second trajectory

Joint	1	2	3	4	5	6
Maximum velocity [rad/s]	0.73	0.78	0.69	1.78	0.76	1.68
Allowed maximum velocity [rad/s] [1]	1.48	1.48	1.31	2.27	2.35	2.35
t [s] at max velocity	2.08	2.95	0.38	2.01	1.39	1.94

Table 4.9: Maximum velocity of each joint for a 4 second trajectory

To achieve the desired 2 second trajectories does not seem possible using the KUKA iiwa. When making the actual system, the maximum speed of the robots should be kept in consideration, when choosing a robot. Choosing a larger robot would also help with this problem, as a larger robot, requires lower joint velocities to achieve the same end effector velocity.

4.5.2. Torque limits

The torque limits of the joints will also be investigated. The KUKA iiwa is rated for lifting objects up to 14 kg. For this system however, a lot or torque is used for pushing on the parcel to make sure there is enough friction. Therefore it is worth it to investigate if the torques as produced by the controller go over the maximal torque for each joint, especially when lifting heavier parcels.

In this test, a parcel with a mass of 7 kg is lifted, and a 4 second trajectory is tracked by the system. The maximum torques of each joint are shown in Table 4.10. Note that only the torque during the actual lifting is measured, so the initial peak in torque that occurs during the initial contact with the parcel is not shown in this table.

Joint	1	2	3	4	5	6
Maximum torque [Nm]	44.2	82.9	48.6	5.3	13.0	0.76
Allowed torque [Nm] [1]	320	320	176	110	40	40
t [s] at max torque	0.40	0.41	1.62	3.31	0.43	0.16

Table 4.10: Maximum velocity of each joint for a 4 second trajectory

Clearly, the robots stay well within their limit for the maximum torque. As for the velocity limits, these limit are a bit arbitrary, as they depend highly on the robot used. It is still a good indication however, that the robots do not use an excessive amount of torque, while lifting heavier parcels.

4.6. Conclusion

From the simulation we can see that the controller can indeed be used for the picking and placing of parcels up to a certain weight. Furthermore, the addition of the adaptive controller can produce a good estimate for the weight of the parcels that are lifted. The addition of the estimated weight to the mass matrix and gravity compensation of the robots improves the tracking accuracy of the system.

From the robustness test we can see that the system can lift parcel up to a certain weight. When the center of mass of the center of the parcel is not in the center of the parcel, but shifted, the system can still produce good results. The maximal weight is not limited by the maximum torque of the joints, but rather by the ratio of the parcel weight to the weight of the robots. Therefore, the controller can most likely be improved to be able to handle heavier parcel. Slowing down the trajectory can slightly increase the weight of the parcels being lifted.

Finally it is shown that the robots stay well within their torque limits during the pick and place task. The robots do however exceed their joint speed limit when tracking the faster 2 second trajectories. These results should only be used as an indication, since the joint limits can vary widely depending on what robots are used. Because of their small size, the KUKA iiwa robot is not the most suited robot to be used when constructing the actual system.

5

Conclusions & Discussion

5.1. Conclusion

This thesis presents an application of a hybrid force position controller to lift parcels using two robot manipulators. It also presents a controller that calculates the desired force on the parcel to make sure the parcels do not slip on the rolling contacts. With a simulation it is proven that the system can pick and place a large variety of parcels.

Since the parcel mass is usually unknown during actual operations, the controller is extended with an adaptive controller. The adaptive controller estimates the weight of the parcel using the forward dynamic equation for robot manipulators. It is demonstrated that the estimator does indeed gives good results for the weight estimation of the parcel, even during motion. Being able to do weight estimation during motion saves time, and will increase the overall speed of the system. With a simulation it is shown that the implementation of the estimated weight improves tracking accuracy of the system.

One of the advantages of the controller is that it only takes Cartesian space coordinates as a trajectory. Therefore, there is no need for the path-planning to be converted to the robots joint space. Al that the path planner needs to do is plan a path for the parcel in Cartesian space. There is no need for complicated closed kinematic chain path planning. For more general path planning, the reach of the two robots should be mapped in such a way, that the path planner can easily distinguish between feasible and non-feasible locations for the parcel. This is not very easy to do because of the complex nature of the kinematics of robot manipulators.

The control scheme presented in this thesis suffices the two goals formulated in the introduction. The first goal mentioned in the introduction is to achieve a low failure rate. When parcel below 5 kg are lifted, the controller rarely fails, even when tracking fast 2 second trajectories. Therefore the system does indeed achieve a low failure rate.

The second goal is that the system should be able to operate quickly. As a benchmark, the system should have an operation cycle of 5 seconds. Since the system is able to track trajectories of only 2 seconds when lifting parcels, an operation cycle of 5 can be achieve using this control scheme.

5.2. Discussion

There are some improvements that can be made regarding the controller in future research. One of the main downsides of the controller is that requires a lot of tuning. Especially the hybrid force position controller has a lot of tuning parameters. Tuning the parameters will have a great influence of the performance of the robot. Especially when using the adaptive controller to estimate the weight of the parcel, the tuning parameters are important. The reason for this is that during the initial moment, when the robots lift the parcel, the weight of the parcel is still unknown. Therefore, the difference between the predicted dynamics and the actual dynamics needs to be compensated by the feedback controller.

Furthermore, some kind of weight estimation of the parcel, before the robots start lifting could help with

the performance of the system. For instance, making a simple estimate based on the size of the parcel.

The way that the required internal force is calculated might also need to change when implemented in an physical system. The controller that calculate the required internal force assumes there is full knowledge about the parcel weight, which is unavailable during the initial lift. The adaptive controller can only estimate the weight of the parcel when it is completely lifted of the ground. Furthermore, the controller assumes it has full knowledge of the friction coefficient, which is not available during actual operation, and highly depends on the material of the parcel. Therefore, the way the required force is calculated should also be changed. For the controller in this thesis, the weight of the parcel that is used to calculate the required internal force is equal to the maximum weight that the robots are able to lift. It is set to 8 kg.

As mentioned in Section 5.1, the controller takes positions in Cartesian space to move to. Therefore it does not required complex path planners that map Cartesian space to the robots joint coordinates. The downside of this, is that it is not possible to move the controller to the desired configuration, as there are multiple robot configurations that bring the end effector to the same position in Cartesian space. For manipulation, it is usually better if the robot is in its 'elbow up' state, where the robots 'elbow' (or the 4th joint of the KUKA iiwa), is above the direct line between the robot base and the end effector. During the simulations run, the robot has never moved to an undesirable configuration, the robot has always been in the elbow up configuration initially when lifting, and it has remained in that position while tracking all trajectories.

5.3. Future research

For further research in the development of a system that can pick and place parcel for a sorting facility, there are couple of main aspect that need to be kept in mind. The first is a selection of the robots. The reach of the system is quite limited, because the reach is limited by both the manipulators. Therefore, a robot with a large reach should be chosen when making a physical system.

The detection of the parcel should also be investigated. For the robots to be able to operate, the position and size of the parcel should be available to them. The most obvious solution would be to have some kind of visual detection system, that uses camera to measure the position, size and shape of the parcels.

A path planner also be should be made. For the controller in this thesis, path planning should be fairly easy, since the controller only needs a path for the parcel in Cartesian space. However, when the planner also needs to stay within the joint limits of the robot, especially the joint speed limits, the path planning becomes a lot more difficult, since it will have to calculate the joint positions of the robot beforehand.

Bibliography

[1] Sensitive robotics lbr iiwa. https://www.kuka.com/nl-be/products/robotics-systems/industrial-robots/lbr-iiwa, 2017.

[2] Farhad Aghili. Adaptive control of manipulators forming closed kinematic chain with inaccurate kinematic model. *IEEE/ASME Transactions on Mechatronics*, 18(5):1544–1554, 2012.

[3] Suguru Arimoto, Morio Yoshida, J-H Bae, and Kenji Tahara. Dynamic force/torque balance of 2d polygonal objects by a pair of rolling contacts and sensory-motor coordination. *Journal of Robotic Systems*, 20 (9):517–537, 2003.

[4] Suguru Arimoto, Hiroe Hashiguchi, and Ryuta Ozawa. A simple control method coping with a kinematically ill-posed inverse problem of redundant robots: Analysis in case of a handwriting robot. *Asian Journal of Control*, 7(2):112–123, 2005.

[5] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2020.

[6] Algemeen Dagblad. Pakketbezorger zijn is meer dan alleen 120 keer busje in, busje uit. `https://www.ad.nl/ad-werkt/pakketbezorger-zijn-is-meer-dan-alleen-120-keer-busje-in-busje-uit~a81eb41f`, 2018.

[7] Z Doulgeri and A Golfakis. Nonlinear manipulation control of a compliant object by dual fingers. *Journal of dynamic systems, measurement, and control*, 128(3):473–481, 2006.

[8] Andrej Gams, Bojan Nemec, Auke Jan Ijspeert, and Aleš Ude. Coupling movement primitives: Interaction with the environment and bimanual tasks. *IEEE Transactions on Robotics*, 30(4):816–830, 2014.

[9] Wail Gueaieb, Fakhri Karray, and Salah Al-Sharhan. A robust adaptive fuzzy position/force control scheme for cooperative manipulators. *IEEE Transactions on Control Systems Technology*, 11(4):516–528, 2003.

[10] Yiming Jiang, Zhi Liu, Ci Chen, and Yun Zhang. Adaptive robust fuzzy control for dual arm robot with unknown input deadzone nonlinearity. *Nonlinear Dynamics*, 81(3):1301–1314, 2015.

[11] Jinoh Lee, Pyung Hun Chang, and Rodrigo S Jamisola. Relative impedance control for dual-arm robots performing asymmetric bimanual tasks. *IEEE transactions on industrial electronics*, 61(7):3786–3796, 2013.

[12] Kevin M Lynch and Frank C Park. *Modern Robotics*. Cambridge University Press, 2017.

[13] Richard M Murray. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[14] Ryuta Ozawa, Suguru Arimoto, Shinsuke Nakamura, and Ji-Hun Bae. Control of an object with parallel surfaces by a pair of finger robots without object sensing. *IEEE Transactions on Robotics*, 21(5):965–976, 2005.

[15] Eric Paljug, Xiaoping Yun, and Vijay Kumar. Control of rolling contacts in multi-arm manipulation. *IEEE Transactions on Robotics and Automation*, 10(4):441–452, 1994.

[16] Stanley A Schneider and Robert H Cannon. Object impedance control for cooperative manipulation: Theory and experimental results. *IEEE Transactions on Robotics and Automation*, 8(3):383–394, 1992.

[17] David Williams and Oussama Khatib. The virtual linkage: A model for internal forces in multi-grasp manipulation. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 1025–1030. IEEE, 1993.

[18] Chenguang Yang, Yiming Jiang, Zhijun Li, Wei He, and Chun-Yi Su. Neural control of bimanual robots with guaranteed global stability and motion precision. *IEEE Transactions on Industrial Informatics*, 13 (3):1162–1171, 2016.