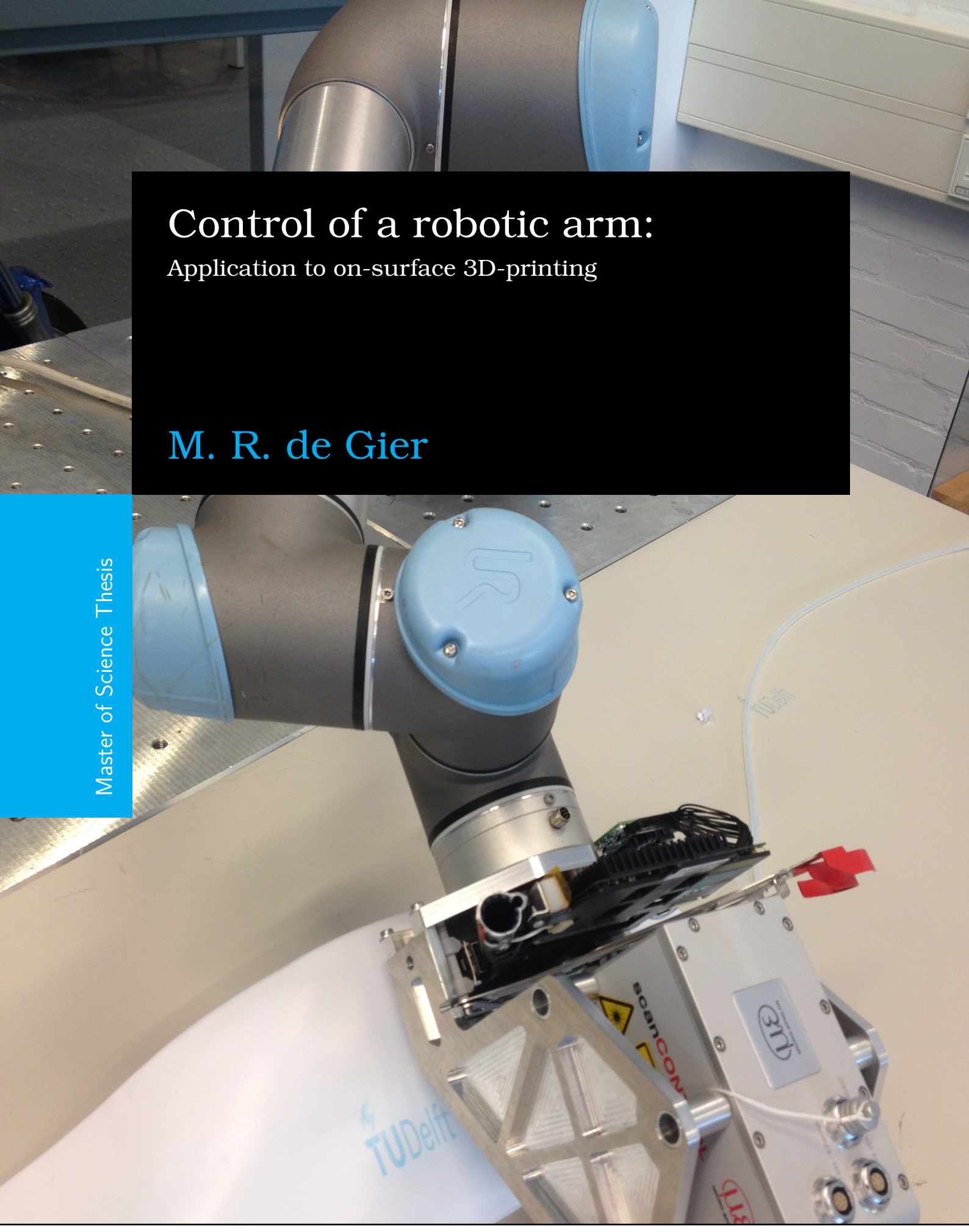


Control of a robotic arm:

Application to on-surface 3D-printing

M. R. de Gier

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Control of a robotic arm:
Application to on-surface 3D-printing

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

M. R. de Gier

April 6, 2015

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

A CANON COMPANY

The work in this thesis was supported by Océ-Technologies B.V. Their cooperation is hereby gratefully acknowledged.

Abstract

3D-printing, also called Additive Manufacturing, is a rapidly developing technique in manufacturing. Nowadays 20% of the printed products is a final product, the rest is used in Rapid Prototyping. It is claimed that in 2020, 50% of the printed output is a final product.

The overall goal of this research is to print 3D-structures on curved surfaces. The print technique used is called Drop-on-Demand and uses a print head with multiple nozzles. The technique enables accurate and fast 3D-prints, but needs a 6-DOF manipulator to be able to print on curved surfaces. For this purpose a 6-DOF robotic arm, the UR5, is used.

In 2013 a working prototype is developed by C.J. Kruit. It was proven that a robotic arm can be beneficial to Additive Manufacturing. Still as a first prototype there is much room for improvements.

The goal of this research is develop a control strategy to be able to print accurately on double curved surfaces. Model Predictive Control is used as a strategy to minimize the error of the print.

In this research also an approach to acquire a local model of the UR5 is developed. Furthermore feasibility of the desired printing path and safety of using the UR5 are topics of this research.

Table of Contents

Acknowledgements

I would like to thank my supervisors Prof. Dr. Ir. Robert Babuška and Prof. Dr. Ir. Jo Geraedts for introducing me to this research topic and for their guidance during this graduation project.

Prof. Dr. Majid Zamani I want to thank for his work as my daily supervisor. Also I would like to thank Ir. Otto Salomons for his useful comments on the project.

Furthermore I would like to thank my supervisors at Océ-Technologies B.V., Ir. Ivan Smits and Ir. Marco Moens for their guidance during my internship. Ir. Sunniva van Ipenburg I want to thank for the helpful brainstorms during troubleshooting.

Delft, University of Technology M. R. de Gier
April 6, 2015

Chapter 1

Introduction

1-1 Background

In the fourteenth century the printing press was an enormous boost for science. The fact that books could now be printed in high numbers, meant that the knowledge could be stored and spread more easily amongst people.

The next revolution in printing has been three dimensional (3D) printing, also referred to as Additive Manufacturing (AM) and Rapid Prototyping (RP). In 1984 C.W. Hull invented Stereo Lithography (SLA) [1] and his invention is regarded as the start of AM. AM techniques are fundamentally different from subtractive manufacturing techniques (e.g. milling, turning and drilling) as they add material instead of removing it. In general the 3D print is manufactured by printing consecutive layers on top of each other until the desired product is formed.

The major benefit of AM is the increase in design freedom. It enables for instance to manufacture hollow products with complex shapes out of one piece. Furthermore one is able to use different materials during printing. It is an sustainable technique as there is in theory no waste material. The absent of moulds and its ability to produce complex shaped products makes the AM especially suitable for one piece production and RP.

One major drawback for AM techniques is the lack of variety of materials that can be used - at this moment -, another is that the techniques are relatively time consuming.

Different techniques for 3D printing have been developed. The most used techniques are: SLA, Fused Deposition Modelling (FDM), PolyJet and Ink Powder Binding (IPB). For desktop printers Formlabs' Form 1 (SLA) and the Ultimaker (FDM) are popular. In the industry Stratasys' Objet Eden 260 (PolyJet) and 3D Systems' ProJet 660 (IPB) are well used.

The development in 3D printing is moving from RP to making final products ready for the market. In 2011 already 20% of the output of 3D printers are now final products, according to Terry Wohlers, who runs a research firm specialising in the field. He predicts that this number will rise to 50% by 2020 [2].

Apart from printing spare parts, scale models and art, the medical industry is now also using 3D printing. Several studies have started to print human body parts. This does not only mean organs [3] but also prostheses. In 2012 LayerWise, specialist in AM, printed a lower yaw bone of titanium [4].

1-2 Motivation

In this research the goal is to be able to print 3D objects, but also 2D prints, on 3D objects using a 6 Degrees Of Freedom (DOF) robotic arm. One could say an extra dimension is added to the already existing 3D printers. Where conventional printers normally have to start the print from a flat surface, the printer of this research can print on already existing objects.

Another major benefit compared to the conventional printers is provided by the use of the robotic arm. The printer is now able to print in an increased workspace, because of the reach of this arm. This is very different from printers like the Ultimaker, where the maximum size of the product is confined by the dimensions of the printer itself. It would also be very interesting to mount the robot arm to a moving platform, so the workspace would become infinite in theory.

A couple of applications are very suitable for the 3D printer developed with the right print head. One of those is to be able to combine 3D printing with other manufacturing techniques. As already pointed out conventional 3D printing mostly needs to start from a flat surface and is relatively slow for large volumes. Because of this, injection moulding is a better option for producing for instance a plastic mug in mass production. However when using the 3D printer from this research one will be able to print one's name on this mug. So the common part can be manufactured by conventional techniques and the customized parts can be printed. This makes customized mass production much faster and cost effective.

Other applications of the printer of this research are to be able to coat or plaster large objects. Also filling cavities and repairing products are new applications very suitable for this new printer.

1-3 Initial Work

This research is based upon the work of C.J. Kruit [5]. The goal of his research was to answer the following research question: Is it beneficial to use a Multiple Degrees of Freedom robotic arm in Additive Manufacturing?

During his research a prototype was built, as in Figure 1-1. This prototype consists of three major parts and self-developed hardware and software to combine these parts. The UR5, a robotic arm developed by Universal Robots is one of these parts. This robotic arm is not specifically made for high accuracy. Also a print head from Océ is used in this set-up. Furthermore a laser scanner from Micro-Epsilon, type scanCONTROL 2700-25, is mounted on the robotic arm in the prototype.

The robotic arm is controlled by Robot Operating System (ROS) which runs on Ubuntu. The drivers of the print head and the laser scanner are running on Windows. This means

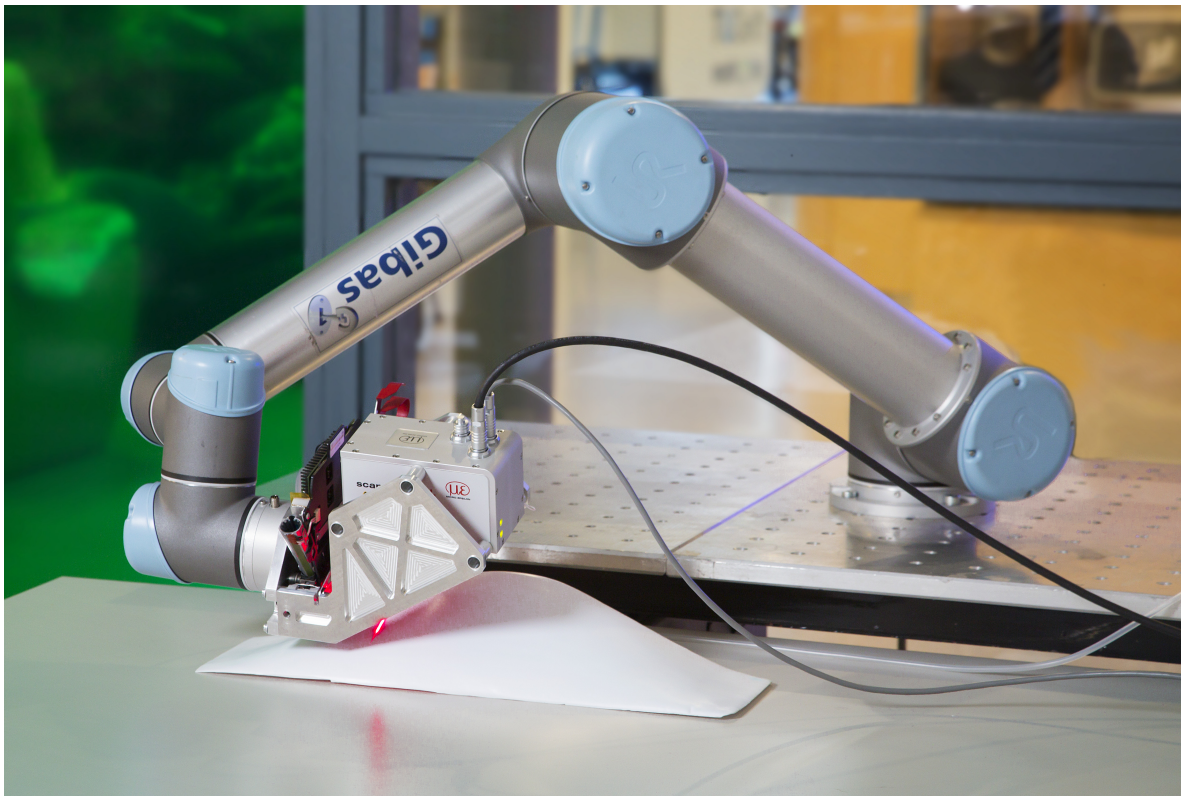


Figure 1-1: The prototype

two computers are used in order to run the whole set-up. The communication between the different drivers and computers is established by three micro-controllers, Arduinos.

The update frequency of the robot arm is 10 Hz, which is quite low for the control tasks. A schematic of the whole system is shown in Figure 1-2 borrowed from [5]. The bottleneck in this system is the UR5-driver, which only runs on Linux. When using the Matlab-driver for the UR5, running on Windows, the whole system can be simplified to having only one PC and one Arduino.

The control of the UR5 in this research is achieved by using five parallel PID controllers. One controller for maintaining the height between the print head and the printed object. One for tracking the printing path in x-y-plane. And one for each of the roll, pitch and yaw angles.

The conclusion of [5] is that it is indeed beneficial to use a Multiple Degrees of Freedom robotic arm in Additive Manufacturing. It is stated that the 3D-printer is more versatile than conventional methods for 3D-printing. Arguments given for this statement are:

- Printing is possible in all directions, so printing time is reduced.
- On-surface printing is possible.
- New building strategies can be developed to minimize warping and to achieve smooth surfaces.

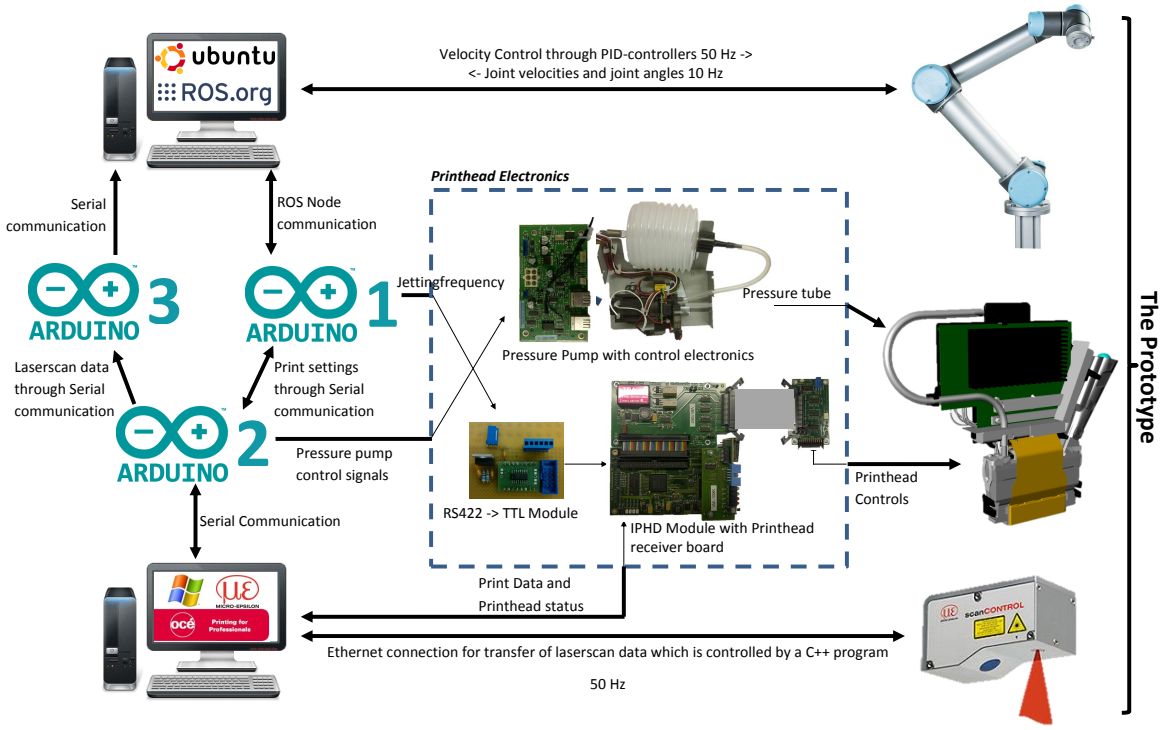


Figure 1-2: Overview of the developed prototype (Source: [5])

The accuracy of the robot is still an issue in his research. The error in x and y-position is larger than 1 mm and increases with increasing velocity of the end-effector. The standard deviation of the position is around 0.1 mm, also significant.

1-4 Goals and Requirements

The overall goal is to print 3D structures on double curved surfaces. To accomplish this developments had to made on many aspects of the whole process. Most of the developments in [5] had to be reconsidered and reversed. For example the circuitous communication software was not suitable for high sample frequencies and thus not for fast and accurate control. But the biggest drawback was the fact that the system did not know beforehand the object on which it would be printing on. This implies that there was no way to point out where there had to be printed on the surface, to alter the image with regard to the curved surface, such that a neat structure was printed and to keep the print head perpendicular to the surface for double curved surfaces instead of single curved surfaces.

An overview of this project in the form of a flowchart is presented in Figure 1-3. A part of the subjects is taken care of in the research of S.C.A van Ipenburg [6]. A summary is given in Section 1-5.

The goals and requirements specifically for this project are:

- Explore the possibilities for the UR5.
- Achieve safe control of the UR5.
- Develop a method to avoid near-singular configurations and collision with itself and other objects.
- Develop a dynamic model for the UR5.
- The controller should cope with as input a trajectory in Cartesian space added with a rotational vector ($[x, y, z, Rx, Ry, Rz]$).
- Achieve a precision of 100 μm for the print.
- All operation should be executable from one device.

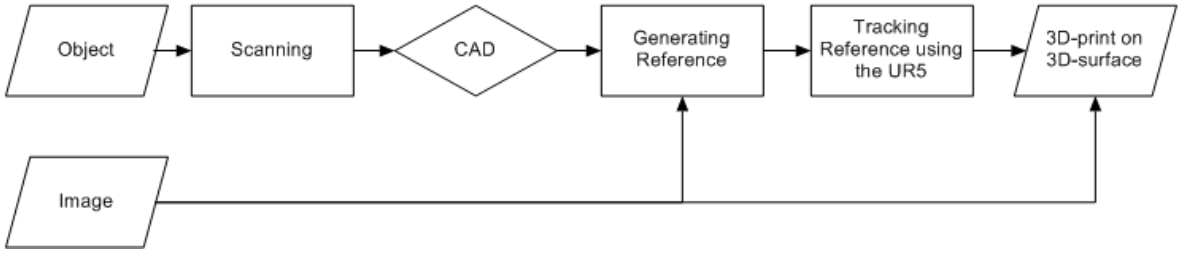


Figure 1-3: Flowchart of the project

1-5 Related Work

This thesis is part of a bigger project as also mentioned in Section 1-4. The overall goal of the project is to print 3D structures on double curved surfaces. In this report the focus is on the control of the robotic arm, while in the research of S.C.A van Ipenburg [6] the focus is on the print strategies and the printing itself.

In this work the conclusion is that it is possible to print on a double curved surface. First the to-be-printed-on object will be scanned with the Epsilon laser scanner my moving the robotic arm. This data combined with the data of the robotic arm will create a reference model. With this model the position in joint space on each time frame can be calculated to print the image or structure on the desired object.

In 2011 Jean-Pierre Gazeau [7] published a paper in which he describes a printer which can cover large slightly-curved surfaces. The printer uses a 5-DOF robot and UV-cured ink. It uses a Cartesian framework and within this framework it can rotate the print head along two perpendicular axes. The printer of Gazeau cannot print 3D objects. The reason that this robot does not need 6-DOF is because with this 5-DOF it can cover the whole surface and the print head does not have to rotate for a good printing result. Due to the construction of the robot with the Cartesian framework and the use of only small angles, the researchers simplify the problems with the inverse kinematics. Also only three distance sensors are used to calculate if the print head is perpendicular to the surface. Due to the construction of the

robot and linearisation of the Jacobian, these measurements can directly be related to the last three joints with linear formulas. This also means that the first two joints can be controlled by an open-loop controller, while the last three joints are controlled in a closed-loop fashion, by a PI controller, using the data from the distance sensors.

There are also several patents where 2D-printing on 3D-objects has been investigated, such as in [8], [9] and [10]. They cover topics very similar to [7], describing methods to print on bowling pins and cans and describing methods to keep the print head perpendicular to the surface. The main difference between their work and this research is that this research makes use of a serial robotic arm in stead of Cartesian frameworks. A robotic arm can print on larger surfaces with more versatility, hence it will prove more difficult to acquire high print qualities.

In a paper of Chen Xinwei [11] from 2011, is written about a 7-DOF robot to print on 3D-surfaces. The main problem tackled in this paper is the height difference between the print head and the surface during a single print. They are not using the versatility of the 7-DOF robot, but instead relying on a very accurate model of the surface. Using this model they can calculate with probability models when a drop of ink has to be jetted, instead of being perpendicular to the surface. Because the print head does not have to face the printing surface perpendicularly, now also steep concave surfaces can be printed. In general this is not possible due to the dimensions of the print head and the danger of collision.

Another research using a 6-DOF robot for printing material is [12]. In this research silver particles are printed in order to build an electrode. The reached precision is very high, between 70 μm and 90 μm. This is achieved by having a very accurate robot arm, which has a repeatability of 20 μm. Also a suspended table is used and a High Definition camera to track the position very accurately.

In 2011 MIT reports of their 3D-printer [13]. It is a robotic arm and the print head can be changed for different applications. The robot can print ABS, but also recycled plastics and concrete. One of the goals of the research is to print in different densities, to achieve more nature-like structures.

1-6 Outline of the thesis

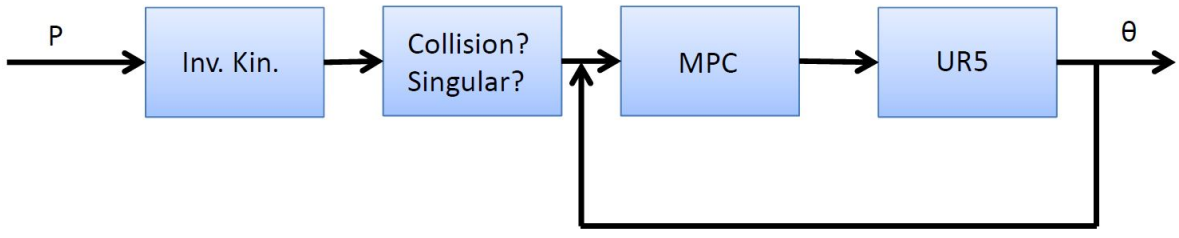


Figure 1-4: Overview of this research

The goal of this research is to track a given trajectory, p in the Cartesian space with the UR5. This must be done such that the print is as accurate as possible. This means that the angle of the joints, θ should be tracked as accurate as possible. To do so the reference in Cartesian

space is converted to a reference in the joint space of the robot, with the use of Inverse Kinematics. After this it is checked if the trajectory is feasible, to check whether there will be no configurations which will cause collisions or are singular. This reference is fed into the Model Predictive Control (MPC) controller to track the reference. To make everything work perfectly a good knowledge of the UR5 is needed. Knowledge such as the specifications, the kinematic model and the dynamic model. A schematic overview of this is shown in Figure1-4. At the beginning of every Chapter it is shown on what part of this scheme the Chapter is elaborating on.

In Chapter 2 the UR5 will be introduced. It will be explained what the specifications of this robot are and how the communication to this robot works. In Chapter 3 the Forward Kinematics of the UR5 will be explained and there will be elaborated on techniques for detecting (near)-singular configurations and collisions. In Chapter 4 a model of the UR5 will be constructed using the Subspace Identification method. Also the time delay of the UR5 will be explained and in this Chapter will be justified why the UR5 can be regarded as a SISO system. In Chapter 5 the Inverse Kinematics of the UR5 will be explained. Next in Chapter 6 the chosen control method, MPC, will be explained and compared to other control methods. In Chapter 7 all the conclusions of the research are summarized. Finally in Chapter 8 some recommendations are made for future work.

Chapter 2

The Robotic Arm: UR5

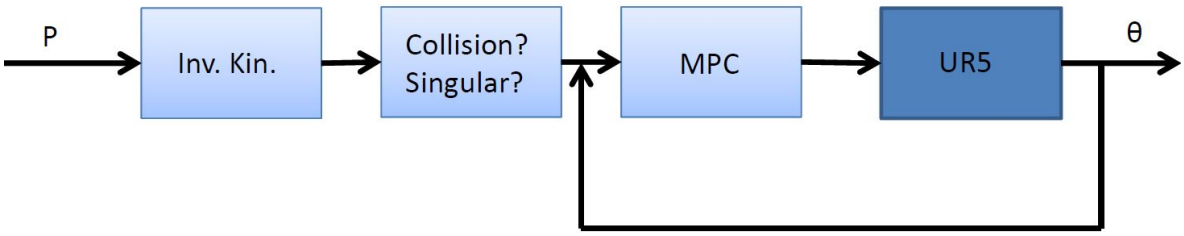


Figure 2-1: In this Chapter there will be elaborated on the UR5

The robotic arm, the UR5, is the key component of this research and therefore it is important to have a closer look at this robot. In this chapter an overview will be given of the capabilities and incapabilities of the UR5. The specifications will be mentioned and the consequences of those specifications will be noted. Furthermore there will be explained the best way of controlling the robotic arm for this research. There are a few methods to control the UR5 all with their own benefits and drawbacks. Another section will be about the safety of the UR5. The UR5 is regarded as a safe robot, therefore a cage is not necessary. Still it can do severe harm when not handled carefully or without the right measures.

2-1 Specifications

The UR5 is a robot developed by the company Universal Robots. Universal Robots is a Danish company and they only sell two products. The UR5 and UR10. The UR10 is somewhat bigger and able to handle a maximum of ten kilograms instead of five. According to the company the key benefits of their robots are that they are light weight, safe and easy to use. The robotic arm is regarded as safe, because it will stop acting as soon as the robot hits an object sensed by a force sensor in one of the joints. Because of this, the robot does not need a safety cage and therefore it makes it ideal to work with as object for this research.

In Figure 2-2 the specifications given by Universal Robots are stated. One important statement from the specifications is the repeatability of 0.1 mm. Although this is within the requirements defined in Section 1-4, other comparable robots, in terms of size and payload, do much better. Both the $IRB1200$ of ABB and the $TX60$ of Stäubli have a repeatability of 0.02 mm. These robotic arms can acquire these values because they are much heavier and stiffer. This also means they are a lot more hazardous and more safety requirements are needed. For example a safety cage. This means these robots are a lot harder to work with in a research environment.

6-axis robot arm with a working radius of 850 mm / 33.5 in

Weight	18.4 kg / 40.6 lbs
Payload	5 kg / 11 lbs
Reach:	850 mm / 33.5 in
Joint ranges:	+/- 360° on all joints
Speed:	Joint: Max 180°/sec. Tool: Approx. 1 m/sec. / Approx. 39.4 in/sec.
Repeatability:	+/- 0.1 mm / +/- 0.0039 in (4 mil)
Footprint:	Ø149 mm / 5.9 in
Degrees of freedom:	6 rotating joints
Control box size (WxHxD):	475 mm x 423 mm x 268 mm / 18.7 x 16.7 x 10.6 in
I/O ports:	10 digital in, 10 digital out, 4 analogue in, 2 analogue out
I/O power supply:	24 V 1200 mA in control box and 12 V/24 V 600 mA in tool
Communication:	TCP/IP 100 Mbit: IEEE 802.3u, 100BASE-TX Ethernet socket & Modbus TCP
Programming:	Polyscope graphical user interface on 12 inch touchscreen with mounting
Noise:	Comparatively noiseless
IP classification:	IP54
Power consumption:	Approx. 200 watts using a typical program
Collaboration operation:	Tested in accordance with sections 5.10.1 and 5.10.5 of EN ISO 10218-1:2006
Materials:	Aluminium, ABS plastic
Temperature:	The robot can work in a temperature range of 0-50°C
Power supply:	100-240 VAC, 50-60 Hz
Calculated Operating Life:	35,000 Hours

Figure 2-2: Specifications of the UR5

In Figure 2-3 the UR5 is drawn with the dimensions of all the links. The robotic arm consists of six revolute joints. For the remainder of the report these joints will be referred to as Base, Shoulder, Elbow, Wrist1, Wrist2 and Wrist3. The names of these joints are also stated in Figure 2-3. The UR5 has a rather standard layout for these types of robotic arms. The Shoulder and Elbow joint are rotating perpendicular to the Base joint. These three joints are connected with long links. In this way the robot has a long reach. The Wrist joints are mainly to manoeuvre the Tool Center Point (TCP) in the right orientation. Other robotic arms with similar layouts often use a spherical joint instead of three revolute joints. Those spherical joints are less prone to self-collision as the solution of the UR5.

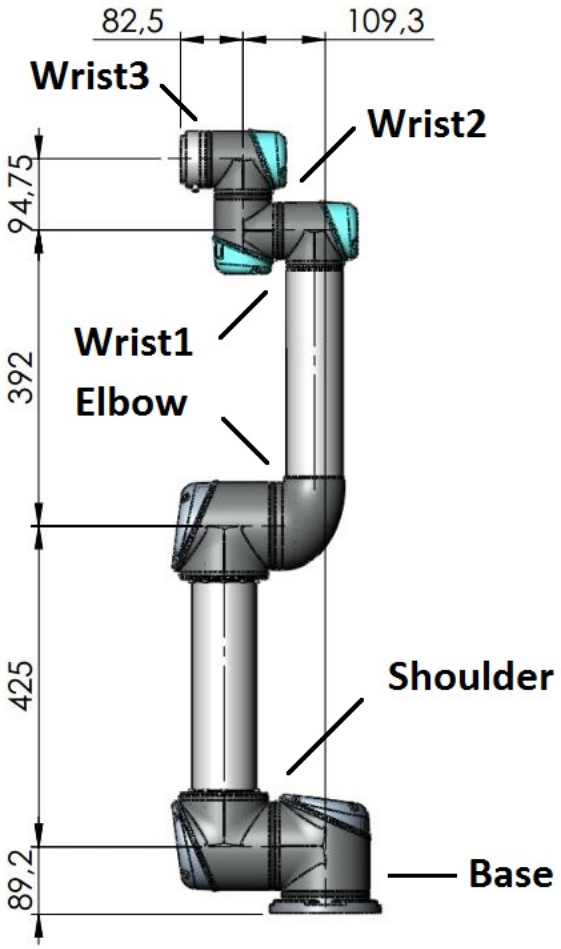


Figure 2-3: Dimensions of the UR5

2-2 Safety

In this section there will be a few words of notion about the safety of the UR5. In general this robotic arm is a safe robot. As already mentioned there is no need for safety shielding according to Universal Robots. This is because the arm will detect an impact larger than 200 N and will go into error mode. Still this will not guarantee that the robot is harmless. Especially the momentum of the robot can do harm. According to the data sheet in Figure 2-2 the arm weighs 18.4 kg (without the payload of 5 kg) and according to Coro [14] the maximal velocity of the TCP is 3 m/s in stead of the 1 m/s stated in the specifications. Also Coro did some tests to measure the impact. It measured forces up to 1500 N. The tests are not completely comparable to impacts with humans as it were impacts of steel to steel. On the other hand the tests were performed at a speed of $0.5m/s$, so they give quite an indication that being hit by the UR5 can be harmful. Besides the problem of hitting people, the robot can also hit itself quite easily, because of the lack of a collision prediction. The UR5 has not an internal model for this purpose and can damage itself. A solution for this problem is proposed in 3-3.

So clearly the robot is not always safe and there are some situations in which extra caution

is necessary and there are safety measures to be taken in such a situations? There are two situations which deserve different measures, namely manually moving the robot with the touch screen or let the arm follow a trajectory in position, for instance with the use of Matlab.

Most of the time when moving the arm manually with the touch screen, one wants to move the TCP in Cartesian space. What can happen in these situations is that the TCP will move through a singular position, as will be further explained in Section 3-2. By doing this, one or more joints will have to go faster than their maximum speed, to follow the set point given. This means the TCP will be lagging the actual position. When the arm is set free from the singular position it wants to keep up with this set point, which will result in a very violent and sudden movement. Another problem to occur is a switch from different configuration during moving the robotic arm. Most of the positions of the TCP can be reached in 8 different positions of the robotic arm. The internal computer of the UR5 will calculate which joint positions are needed for the corresponding position in Cartesian space. But in some positions it will switch from one configuration to a totally different one. This is illustrated in Figure 2-4. To reach the same position, the joints of the 3DOF robotic arm needs to change much. The Base joint reverts completely. This situation is a again a very violent and sudden movement for the UR5. There is not much which can be done to prevent those situations from occurring, besides being very careful when moving the robotic arm in Cartesian space. When doubting it is possible to lower the velocity of the arm for manual movement. It is always better to move the robot by adjusting the joints, but this is not always wanted.

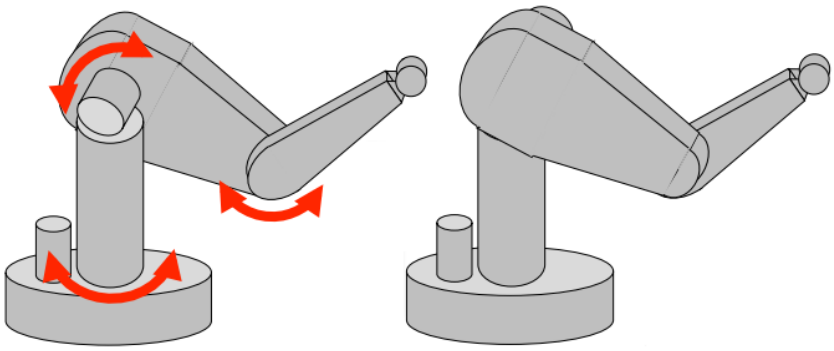


Figure 2-4: An example of two different configuration for the same position of the TCP in Cartesian space for a 3DOF Robot

The case when following a trajectory in position is very different. In this research the robotic arm is controlled by Matlab. The reasoning behind this will be further explained in Section 2-3. With the use of Matlab it is easier to program extra safety programs to make sure nothing will go wrong. But even then it is important to keep the red emergency stop button within reach and people and stuff should stay out of the reach of the UR5. This is especially the case when using a new, not tested before, program. In this case the controller should not be too aggressive. It is better to gradually increase the values of the controller.

When having a trajectory, it is possible to do some calculation beforehand. This can be before starting the whole printing process, or even a few samples in advance. These calculations are to see if the robot will get into a singular configuration or to see if the robot will collide with itself or the environment. This will be the topic of Section 3-2 and 3-3.

The last safety measure is a program which checks every time step if all the joints are within the error bounds. When this is not the case, the program will go into error mode and stop the robot. This will prevent it from colliding or being able to do unpredictable and hazardous movements.

2-3 Communication with the UR5

The UR5 will be controlled by commanding the velocities of the individual joints of the robot by a driver in Matlab using URControl. In this section it will be explained why these methods of controlling are best and therefore chosen.

In order to move the robotic arm to a position the individual joints or the position of the TCP can be controlled. Controlling the TCP can lead to unpredictable situations, as explained in Section 2-2. One is dependent on the calculations for the inverse kinematics the internal controller will perform. This can lead to very high speeds and accelerations for the individual joints and therefore lead to a shut-down of the system. By commanding the individual joints instead this can be better controlled, resulting in a safer system. It is also never sure in which configuration the robot will be during the tracking of a reference in position for the TCP. This means the checks you like to perform beforehand, collision detection, checking the feasibility of the path and checking if there will be singular configuration, cannot be performed. These major drawbacks lead to the conclusion that the robot should be controlled by commanding the individual joints. It leads to a more complicated program, because the inverse kinematics problem should now be solved without the help of the internal computer of the UR5. This problem is discussed in Chapter 5. The benefit is that the robot will be very predictable and therefore much safer to use.

No outer loop should be needed when controlling the position. But the driver sends only one setpoint to the internal controller of the UR5, while it needs one or more ahead-setpoints for a smooth trajectory. When providing the UR5 setpoints it will try to match the speed profile as shown in Figure 2-5. A maximum speed and acceleration can be set, but the UR5 will always try to follow the speed profile and therefore stop at every setpoint. This has as a result that the robot will have a very jittery behaviour and will lag most of the time. Following the speed profile is no problem for pick and place problems, but it is not suitable for tracking problems. Moreover it is hard to control the velocity of the joint as the UR5 jumps from setpoint to setpoint without considering time. Using software calculating all parameters for the given trajectory one could get quite good results, but besides the fact that this is expensive software it will still not ensure satisfactory timing. This means it is chosen to command the UR5 by the velocity commands. There will therefore be a drift in position over time, but an outer loop can deal with this drift. This problem will be dealt with in Chapter 6.

Most users of the UR5 will not bother to control the robot by an external computer. They will use the teach pendent of the robot to program it. In Figure 2-6 this situation is shown in a scheme. The GUI on the teach pendant, PolyScope, will communicate with the controller of the UR5, URControl. This controller will communicate via the low-level controller in C-API to the robot via ethernet to send and retrieve information. In the research of C.J. Kruit [5] the UR5 was controlled by ROS. This operating system works in Linux and replaces the PolyScope in the scheme in Figure 2-6. ROS is very useful as it has a large base of users and

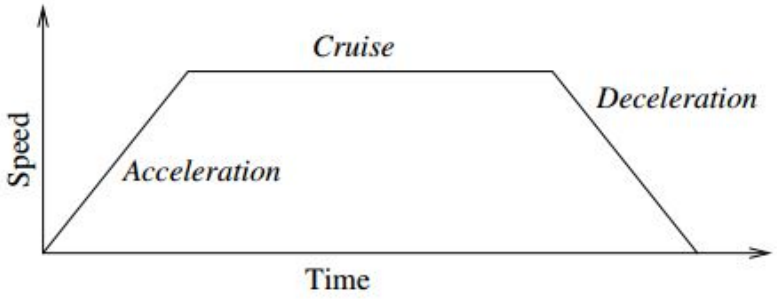


Figure 2-5: The speed profile of a joint of the UR5 when using the position command (Source: Universal Robots)

therefore there are a lot of libraries and forums to solve problems. But as it is the goal to have the whole printer working on one computer and other drivers, for the print head and laser scanner, only work on Windows, ROS is not a suitable option. Instead a driver in Matlab, developed by Delft Center for Systems and Control (DCSC) is used. This driver still replaces PolyScope in all its functionality. The only drawback is that unlike ROS, this driver does not have many libraries and therefore many programs have to be developed.

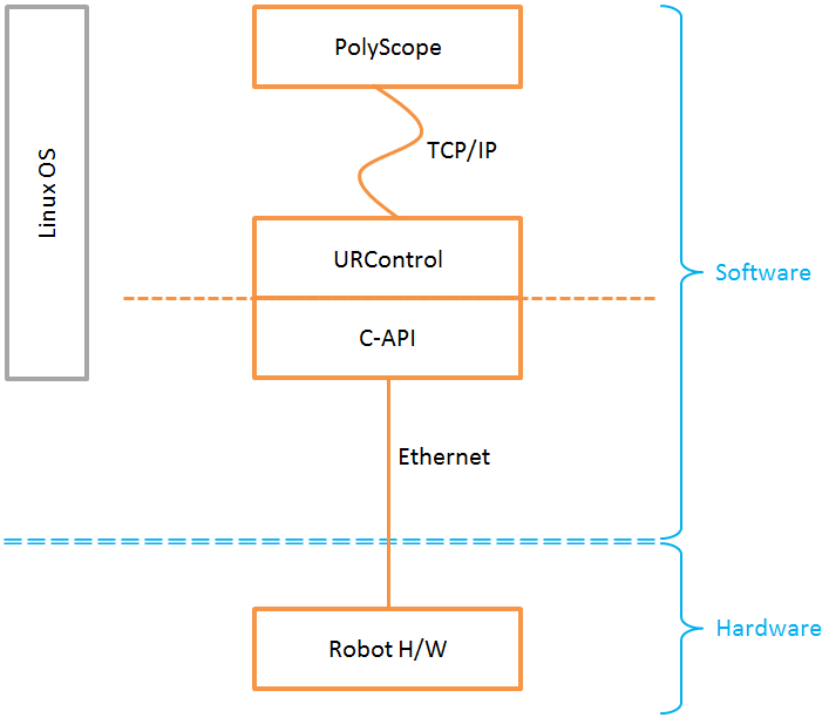


Figure 2-6: Overview of the different layers of control in the UR5 (Source: Universal Robots)

The Matlab Driver can command the UR5 using only command lines in Matlab or to use Simulink. Chosen is not to use the Simulink as it has a worse performance than just using the command-lines. An example of bad performance in Simulink is shown in Figure 2-7. Wrist2

should follow a ramp reference, but around $0.3s$, $0.6s$ and $0.9s$ strange jumps occur. This behaviour is not seen when taking a close look at the joint itself, nor is it dependent on most settings. The jump will always take place at that particular moment. This behaviour is not present when using m-files. Moreover there is not really a need to use the Simulink toolbox. Simulink excels when simulation of dynamic continuous time systems are required and this is not the case for the control of the robot.

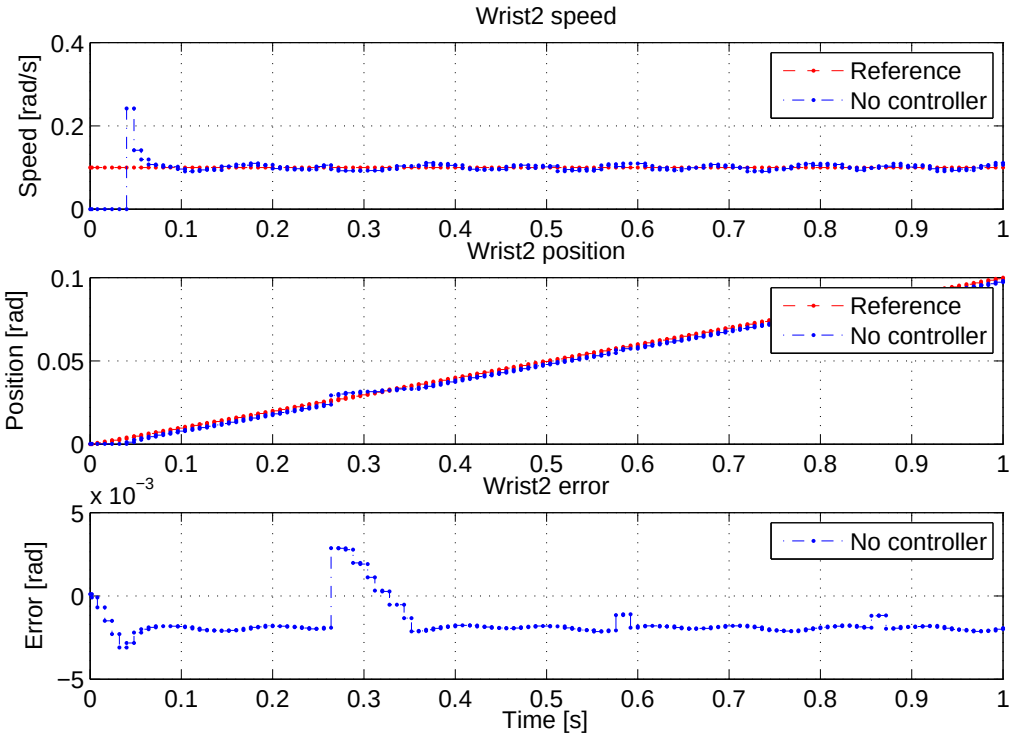


Figure 2-7: An example of strange behaviour in Wrist2 in Simulink

Besides controlling the robot by URControl, via Matlab or ROS one could also choose to use the low-level controller, C-API. This controller can be compiled on the robot self. This is possible because the robot is driven by an industrial Linux-type operation system. Because this controller is more low-level than URControl the latency will be reduced. Which is desirable as explained in Section 4-2, though it is hard to say how much. No communication with an external pc is needed. And due to the lack of I/O-ports for real-time applications with high bit-rates it is hard to communicate with other systems. Also this method is very bare-bone and without testing it is unknown what safety-systems are still in use and which not. The latter two reasons are relevant enough not to use C-API for commanding the robot.

2-4 Conclusions

There are a few methods to command the UR5. The main methods are via ROS using the URControl, via Matlab using the UR control and via C-API. The chosen method is via Matlab as all other programs are also developed in Matlab and this ensures fast communication

between the programs. Within the Matlab Driver no Simulink is used and the velocities in the joint space of the robot are commanded. As this gives the best performance and is safest. The robotic arm can be controlled at a maximum sampling frequency of 125 Hz.

Chapter 3

The Workspace of the UR5

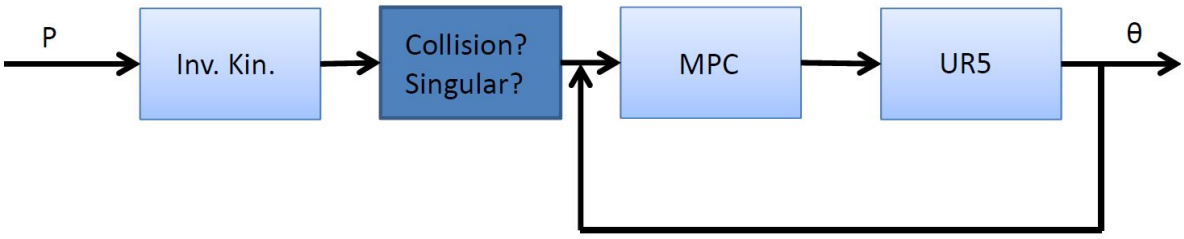


Figure 3-1: In this Chapter there will be elaborated on collision detection and singular configuration detection

One of the benefits of the use of a robotic arm compared to a conventional 3D printer, such as the Ultimaker, is its workspace. The UR5 is not confined to a box space but can reach far further. In Figure 3-2 the workspace of the UR5 is depicted. The green area is the area where the TCP can reach. The outer diameter of the sphere is 1.7 m and the diameter of the unreachable cylinder is 0.3 m. But there are more unreachable configurations. In some positions the robot will suffer from (near)-singular configuration, explained in Section 3-2. Also the robotic arm can collide with itself or the environment, this will be the topic of Section 3-3. But first in Section 3-1 will be elaborated on the mapping from joint space to Cartesian space, before even being able to calculate if there will be any singular configuration or collision.

3-1 Forward Kinematics

Forward kinematics is the mapping from the joint space to the Cartesian space, in this particular case the Cartesian position of the print head.

$$\begin{bmatrix} p^j \\ 1 \end{bmatrix} = H_j^0 \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \tag{3-1}$$

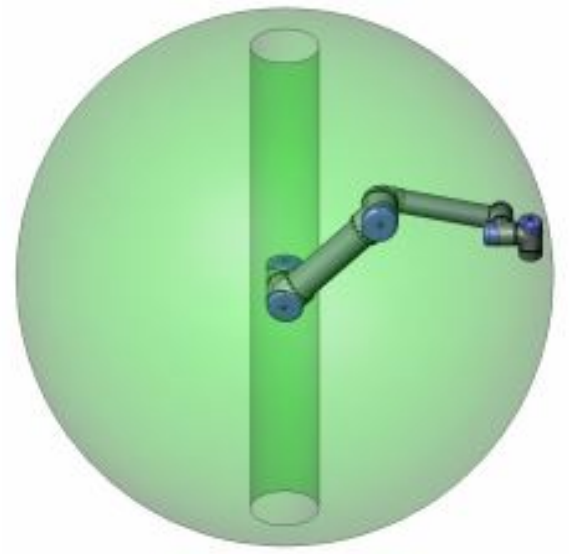


Figure 3-2: The workspace of the UR5 (source: Universal Robots)

$$H_k^0 = H_j^0 H_k^j \tag{3-2}$$

$$H_6^0 = \prod_{i=1}^{6} H_i^{i-1}(\theta_i) \tag{3-3}$$

$$H_i^{i-1}(\theta_i, d_i, a_i, \alpha_i) = R_z(\theta_i)T_z(d_i)T_x(a_i)R_x(\alpha_i) \tag{3-4}$$

The transformation matrix H_j^i is the matrix mapping point p in reference frame Ψ_i into Ψ_j [15]. For a robotic arm it is convenient to begin with the base frame Ψ_0 and start from the zero vector in xyz, as shown in (3-1). Mapping into other reference frames can be easily achieved by multiplication of the transformation matrices as shown in (3-2). Every homogeneous transformation matrix is constructed with the position of the joints of the UR5 and represent the transformation from joint $i-1$ to joint i. Where θ_1 is the Base joint and θ_6 Wrist3. Eventually a mapping from the base frame to the position of the TCP can be constructed as shown in (3-3). These transformation matrices can be easily constructed conform to the Denavit-Hartenberg convention [16] and constructed as in (3-4). These Denavit-Hartenberg parameters are known for the UR5 and listed in Figure 3-3.

$$H_7^0 = H_6^0 H_7^6 = H_7^0 \left[\begin{array}{c|c} & 0 \\ I & d_y \\ & d_z \\ \hline \mathbf{0} & 1 \end{array}\right] \tag{3-5}$$

The print head is rigidly connected to the TCP and therefore this transformation matrix can be extended with a translation matrix to obtain the transformation matrix from the base frame to the position of the print head. In (3-5) is shown how. In this project, $d_y = 71.7\ mm$ and $d_z = 39.5\ mm$.

	$\boldsymbol{\theta}$ $[rad]$	$\boldsymbol{a}$ $[m]$	$\boldsymbol{d}$ $[m]$	$\boldsymbol{\alpha}$ $[rad]$
Joint 1:	0	0	0.08920	$\frac{\pi}{2}$
Joint 2:	0	−0.42500	0	0
Joint 3:	0	−0.39243	0	0
Joint 4:	0	0	0.10900	$\frac{\pi}{2}$
Joint 5:	0	0	0.09300	$\frac{-\pi}{2}$
Joint 6:	0	0	0.08200	0

Figure 3-3: The Denavit-Hartenberg parameters for the UR5

3-2 (Near)-Singularity

Singularity is a common problem in robotics. When a robot is in a singular configuration it cannot track the intended trajectory. This is something to be avoided. But when does singularity occur and what does this mean physically for the robot?

$$v = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} = \begin{bmatrix} J_P \\ J_\omega \end{bmatrix} \dot{\theta} = J_G \dot{\theta} \tag{3-6}$$

$$\dot{x} = \begin{bmatrix} \dot{p} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} J_P \\ J_\phi \end{bmatrix} \dot{\theta} = J_A \dot{\theta} \tag{3-7}$$

$$J_A = \frac{\partial k(\theta)}{\partial \theta} \tag{3-8}$$

The transformation matrix from the joint velocity space to the end effector velocity space is called the Jacobian. There is a difference between the geometric Jacobian shown in (3-6) and the analytic Jacobian shown in (3-7). While ω represents the angular velocity $\dot{\phi}$ represents the change of orientation of the end-effector frame ϕ. ω and $\dot{\phi}$ are in general not the same.

The analytic Jacobian is generally calculated by partial differentials as in (3-8) using $x = k(\theta)$.

$$J_G = \begin{bmatrix} Z_0 \times (o_N - o_0) & \cdots & Z_{i-1} \times (o_N - o_{i-1}) & \cdots & Z_{N-1} \times (o_N - o_{N-1}) \\ Z_0 & \cdots & Z_{i-1} & \cdots & Z_{N-1} \end{bmatrix} \tag{3-9}$$

The geometric Jacobian for a robot with only revolute joints can be constructed by (3-9) [17]. Where Z_i are the first three elements of the third column of transformation matrix H_i^0 and o_i are the first three elements of the fourth column of H_i^0.

When the Jacobian loses a rank, singularity occurs. There are two types of singular configurations for robotic arms, boundary singularities and internal singularities [18]. Boundary singularities are caused when the robot is asked to move outside his workspace. Typically this happens when the robot is stretched out to its full extent. It then loses freedom to move in every direction. Internal singularities are generally caused when two or more axes of the robot are aligned. In this case an action of one joint can be cancelled by another joint. So there are infinite possibilities for the movement, leaving the action undetermined. An example is shown in Figure 3-4. This is also the case for the UR5. The Shoulder, Elbow and Wrist1 are always

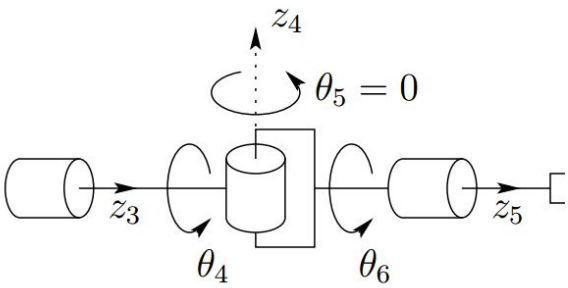


Figure 3-4: An example of an internal singularity

aligned and this is fine. But when Wrist3 also aligned, four joints will do the movement which could be done by three joints. And so a singular configuration is established.

Near-singularities pose also a problem when inverting the Jacobian in (3-6) to the result of (3-10). With near-singularities, when the determinant of the Jacobian becomes small, the inverted Jacobian will become large. This means that for a small movement of the end-effector some joints have to perform a large movement. This means that the velocity of the joint operates outside its physical limit, hence the end-effector will not be able to move according to the given reference trajectory.

$$J_G^{-1} v = \dot{\theta} \tag{3-10}$$

For the UR5 the internal singularities are the most important to deal with, because these are more likely to interfere with the printing trajectory. Boundary singularities are more easily to detect and avoided by staying away from stretching the robot to its full extent. Internal singularities can happen anywhere within these boundaries.

In literature some examples can be found where the authors map the workspace of different robot arms, in [19], [20] and [21]. But the methods are mainly not generic and only to be applied for that specific robot arm. Also [22] and [23] provide novel methods to find singular configurations, but still work needs to be done to design a method to find configurations which are near-singular.

The method used in [5] is not suitable in any case. The solution for the near-singularity problem was as follows:

1. Detecting a near-singular configuration using the condition number of the Jacobian
2. Freeze at that time instance the joint velocity
3. Continue to follow the trajectory as soon as the robotic arm is out of the near-singular configuration.

The benefit of this approach is that the robot will continue to act even when entering a near-singular configuration. But this behaviour is unpredictable without simulation. It might be that the robot will eventually collide or reach the end of its reach. Secondly for the application of printing it is highly undesired that the print head deviates from the trajectory.

A better approach and the one used in this research is to look at the consequences of the singular configurations and direct the solution to that. When entering a near-singular configuration, it means that one of the joints has to have a higher velocity than possible, namely 3.4 rad/s. As the trajectory is calculated in joint space beforehand it is easy to note this. When this happens the program will inform the user. The solutions to avoid this near-singular case is to slow down the printing process. Or by finding an alternative trajectory for the print, as suggested in Section 8-3.

3-3 Collision

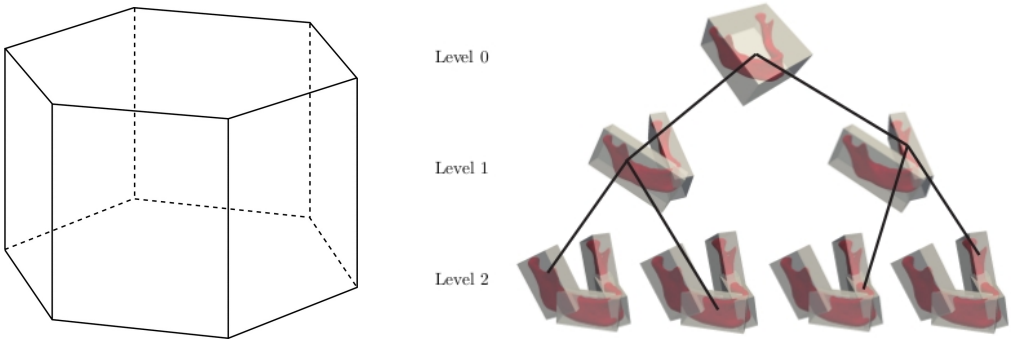
The robotic arm can collide into three different objects. The UR5 can collide into itself, scanner and printer, it can collide into the environment and it can collide into the print. As an assumption the environment will always keep the same position, the laser scanner info can give precise information on the position. The arm is always moving thus the information of the position of the arm will keep changing. The print will also move in time, as it will become bigger in time. Also with the print the laser scanner can give updated information about the position of the print. But it can also be predicted.

The UR5 does not have a function in which it knows in what configurations it will collide with itself. And of course also not when it will hit the print or environment. This should be developed to ensure safe movement.

The easiest way is to confine the joints to a minimum and maximum angle in which it is sure there will be no collision. For instance the Base joint will be free to move every angle. But the Shoulder has to move between 0 and π to avoid collision to the table or pedestal. This approach is very simple to implement, but it will limit the movement of the robot very much. Some configurations within the workspace of the UR5 can be reached without collision, but by limiting the joints it is not a configuration which can be reached. In other words this method is very conservative.

Two other approaches are widely used in the gaming industry, Separating Axis Theorem (SAT) [24] and Gilbert - Johnson - Keerthi (GJK) [25]. Both these methods make use of the Oriented Bounding Box (OBB) [26]. These OBB's are boxes to wholly envelop the difficult shaped object. And contrary to normal bounding boxes, OBB's can be oriented in space, thus the faces are not perpendicular to the Cartesian axes. There is some freedom in choosing how to envelop the objects. It is possible to choose other shapes. The only requirement is that the shape is convex. Also as the methods almost increase exponentially with the number of points, the OBB stays the simplest shape. Still the hexagonal prism, as in Figure 3-5a, is a good alternative for the links of the robotic arm, as it wraps the cylinder more tightly. Another extension to the OBB is the subdivision into multiple smaller OBB's. It is necessary to have a convex shape to envelop an object. But sometimes it is not possible to wrap this shape tightly around the object, due to a weird geometry or protrusions. Then it is a good idea to divide the shape in multiple smaller OBB's. In Figure 3-5b a yaw bone is divided in four smaller OBB's.

SAT is a method used in computer games, but also robotics, to detect or predict collisions between different bodies. The method uses the principle that if two convex shapes can be separated, by a line in 2D or by a plane in 3D, they do not collide at all.



(a) A hexagonal prism **(b)** Division in multiple OBB's for a lower yaw bone

Figure 3-5: Extensions to OBB's

In this section it will be clearly explained how this method works in the 2D case and later it will be extended to the 3D case.

As explained earlier a line must be found to separate two convex shapes. It is a little bit extensive to check every line, therefore there is a routine to speed up this search. It is enough to check whether this line is parallel to a face of the convex shapes. This means that for two boxes as in Figure 3-6 only four orientations of this line have to be researched.

The easiest way to check if this line exists, is to check if the projections of the points of the convex sets on a vector can be separated. This vector is perpendicular to a face of one of the convex shapes. As shown in Figure 3-6 the is a vector constructed, perpendicular to the right and left face of the left square. The points of both convex sets projected on this vector can be separated, thus the two sets are not colliding. Finding one line which separates the two sets is enough to prove that the two objects are not colliding.

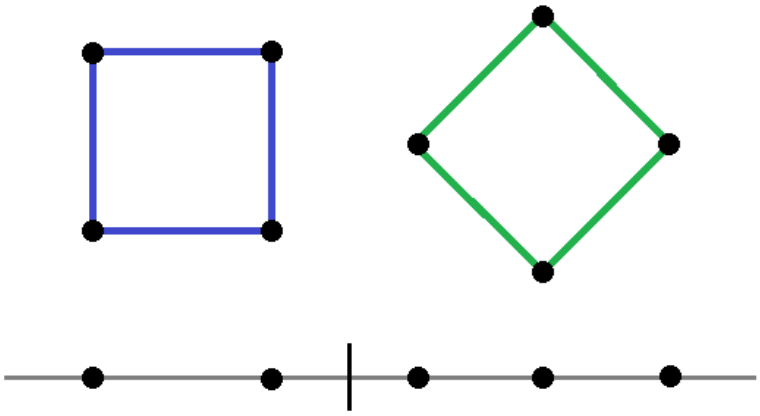


Figure 3-6: Example of the SAT method in 2D

For the 3D case the method works for a major part the same. Instead of a line, now a separating plane has to be found. Still this plane can be found in the same way as the separating line. If the projection of the points of both sets on a vector can be separated, both objects are not colliding. Again not an infinite set of vectors has to be checked. Most of the time only the vectors perpendicular to the faces of the convex sets have to be checked. In

case of OBB's, this means only six vectors have to be checked. But there are cases, where this is not enough. In Figure 3-7 there is a case in which the objects are not colliding, but they can not be separated by a plane parallel to one of the faces of the objects. But they could clearly be separated by a plane. This plane is perpendicular to a vector which is the cross product of the direction of the edges which are almost colliding. And this is the solution for this problem. Besides the faces, also the cross product of the direction of the edges has to be checked. This means for OBB's that in total fifteen vectors have to be checked.

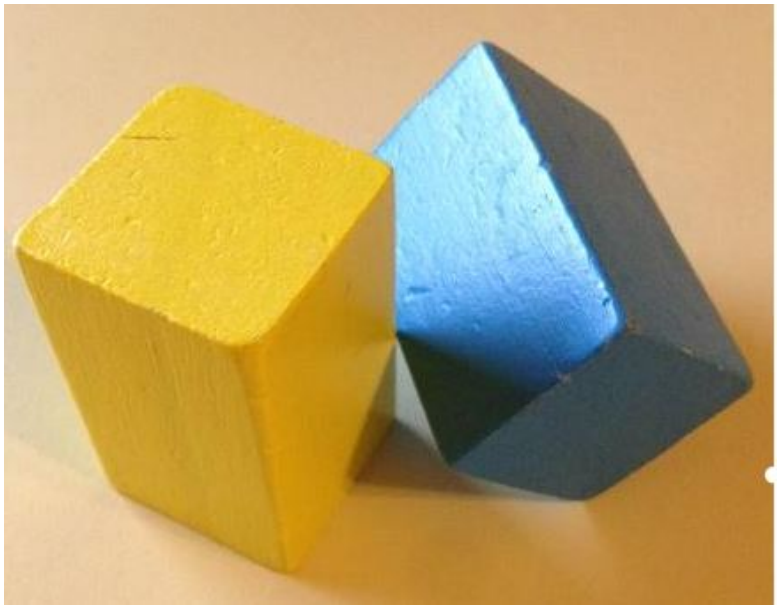


Figure 3-7: Example of the SAT method in 3D

Also GJK is a method to detect collisions. And just like SAT this method is widely used in the gaming industry and also finds its way into the field of robotics.

The method is easy to understand and works the same for in 2D as in 3D. Therefore only the 2D case will be explained, to keep things clear.

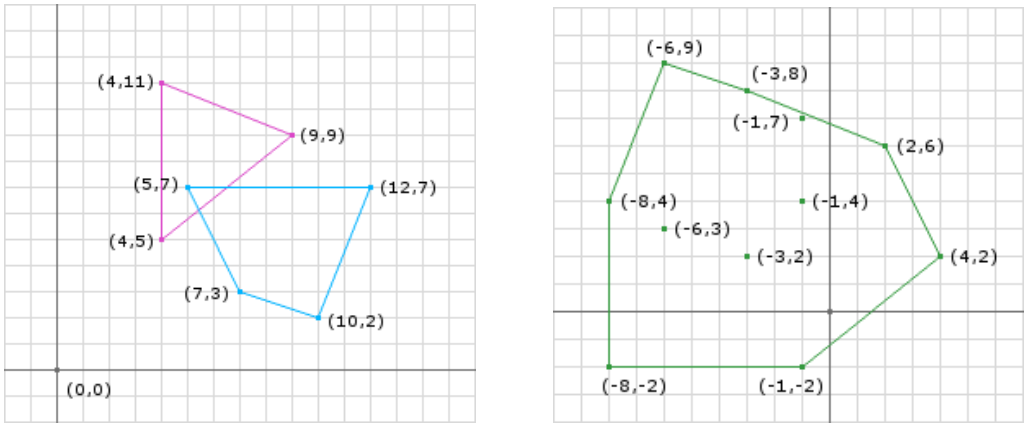
$$A + B = \{a + b | a \in A, b \in B\} \tag{3-11}$$

$$A - B = \{a - b | a \in A, b \in B\} \tag{3-12}$$

The key in this method is the Minkowski Difference. The Minkowski Difference is actually the Minkowski Sum, but for clarity it will be referred to as Minkowski Difference.

The Minkowski Sum is the sum of every point in set A with every point in set B (3-11). Therefore the Minkowski Difference is the subtraction of every point in set B of every point in set A (3-12). This number increases quite rapidly. In the 3D case when using OBB's the number of points in the set required from the Minkowski Difference is 64.

In Figure 3-8 an example is shown of the principle of this Minkowski Difference. In Figure 3-8a two colliding convex sets are shown. In Figure 3-8b the Minkowski Difference of those two sets is taken. The key property of the Minkowski Difference for this purpose is that the



(a) Two convex sets **(b)** The Minkowski Difference

Figure 3-8: Example of the Minkowski Difference for two convex sets (source [27])

convex hull of the Minkowski Difference will contain the origin when the two convex sets are colliding. This is also clearly the case in Figure 3-8.

After taking the Minkowski Difference, the GJK algorithm efficiently decides whether the origin is in or out the convex hull. Until this is decided it cannot be said with certainty if the two objects are colliding.

Both methods, SAT and GJK, are good options for collision detection. SAT has as advantage over GJK that only one separating plane has to be found for deciding that the objects are not colliding. The GJK method needs to complete the whole routine before that can be decided. The advantage GJK has over SAT is that it can handle quite easily complex convex hulls. But as only OBB's are used, or at least simple shapes for the UR5, this benefit is not important for the research.

3-4 Conclusions

Using the Denavit-Hartenberg parameters a kinematic model could be derived of the UR5. This model is extended with the print head and the laser scanner.

When the reference in the joint space is calculated it will be checked whether is lies in the workspace of the UR5. By checking if the velocity of the joint does not exceeds the maximum velocity of the joint, it will be prevented that the UR5 will suffer from near-singular configurations. The solution to this is to slow down the movement of the robot in this part of the workspace. Or by changing the position of the object to be printed on.

It is also checked whether the robotic arm will collide into itself. At the moment this is done by restricting the joints to exceed predetermined angles. This is a very conservative method and it is certain that workspace of the UR5 where no collision takes place is restricted by this method. A proposed method is the SAT method. When the UR5, the environment and the print will be enveloped by OBB's, SAT proves to be very fast and efficient.

Chapter 4

Modelling of the UR5

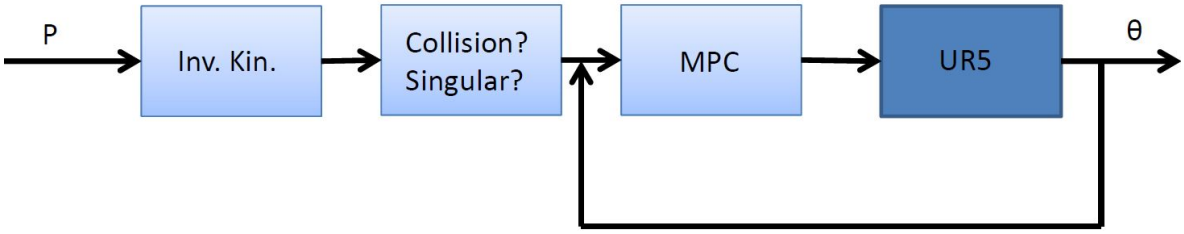


Figure 4-1: This Chapter is elaborating on the dynamic model of the UR5

4-1 Problem statement

As explained in Chapter 2 the robot will be controlled by commanding the velocity for the joints of the UR5. The problem is that, because of this method of commanding, there will be a drift in the position for the joint and therefore also in the position of the end-effector. The solution is to construct a loop with an additional controller around the internal controller of the UR5, as depicted in Figure 4-2. Where r^* is the reference of the desired trajectory and r is the reference given to the internal controller of the UR5.

There is a lot of freedom in designing this controller, this will be elaborated in Chapter 6. It can be decided to design a model free controller for the UR5. For instance a PID-controller. But as explained in Chapter 6, one might want to use a more sophisticated control method like Model Predictive Control (MPC) for better performance. Also when tackling the problem of the time delay, as explained in Section 4-2, it is necessary to have a model of the system. Furthermore a model of the UR5 is needed for the construction of an observer, like a Kalman Filter.

In this Chapter the magnitude of the time-delay in the system is shown. Also the effect of this time-delay will be explained and how to tackle this problem. Furthermore the choice

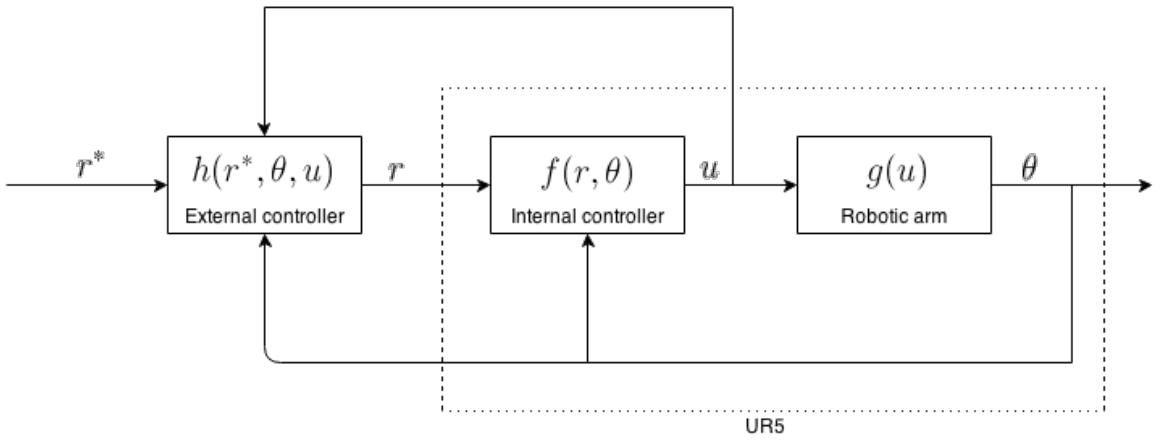


Figure 4-2: A schematic view of the additional control loop

for SID will be explained and under what conditions it is used. Also the model will be validated. Last in this Chapter it will be explained if and under what conditions the UR5 can be regarded as a Single-Input Single-Output (SISO) system instead of a Multiple-Input Multiple-Output (MIMO) system.

4-2 Time Delay

Every control system has some delay between sending a command and receiving data back again. Ideally this time is shorter than the sampling time. In this case there is no delay in the system. Unfortunately this is not the case for the UR5. In Figure 4-3 it can be seen that the Wrist2 Joint takes 5 times the sampling time to react to the step input at a sampling frequency of 125 Hz.

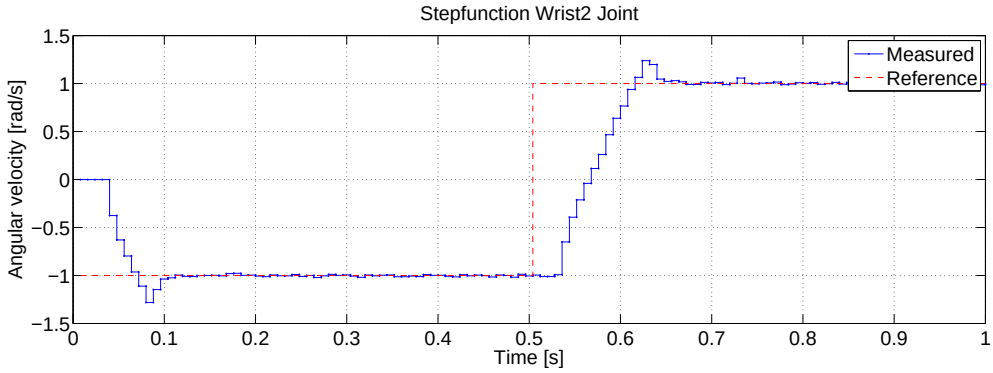


Figure 4-3: The delay for the UR5 at 125 Hz

It is unknown what causes this delay, but it is independent of all parameters which can be manipulated while commanding the robot. This means that this delay is also independent of the sampling frequency and will be around 0.03 seconds. Fortunately the delay is equal in all the six joints. This means that all the joints will be in sync and therefore not affect the

position of the end-effector. In short, the end-effector will be 0.03 seconds later on exactly the right position.

Delays are common in robotic systems, but nevertheless they have to be dealt with. The delay will narrow down the bandwidth of the system and therefore it cannot follow the reference trajectory within the error-bounds. The delay can be dealt with by choosing a clever algorithm or by commanding the robot in a more direct method. This direct method is described in Section 2-3.

$$\bar{G} = Gz^{-k} \tag{4-1}$$

$$H = \frac{CG}{1 + CG} \tag{4-2}$$

$$\bar{H} = Hz^{-k} \tag{4-3}$$

$$\bar{H} = \frac{\bar{C}\bar{G}}{1 + \bar{C}\bar{G}} = \frac{CG}{1 + CG} z^{-k} \tag{4-4}$$

$$\bar{C} = \frac{C}{1 + CG(1 - z^{-k})} \tag{4-5}$$

A method to increase the bandwidth is to make use of a Smith Predictor. The idea of a Smith Predictor is to add an extra loop in the control scheme with a copy of the system as in Figure 4-4.

The system G with k steps of delay forms the new system $\bar{G}$ as in (4-1). This system $\bar{G}$ is unwanted due to the delay, but the delay cannot be taken away. So the Smith Predictor has to handle it. Without the delay the wanted transfer function would be as H in (4-2). But system G is unobtainable and therefore H is not possible to achieve. The solution to this is to define $\bar{H}$, which is transfer function H with a pure delay of k samples (4-3). This is the desired transfer function with only a pure delay and therefore the delay will not interfere in the control loop. To achieve this $\bar{C}$ has to be calculated. To achieve this one should equal the delayed desired system to the transfer function with the actual system, $\bar{G}$ as in (4-4). This gives as a result the controller $\bar{C}$ defined as in (4-5).

By doing this the result is transfer function $\bar{H}$ with a pure delay, instead of the delay action interfering with the controller. The downside of this method is that the performance is sensitive to the estimation of $\hat{\mathrm{G}}(\mathrm{z})$. For longer experiments the output of $\hat{\mathrm{G}}(\mathrm{z})$ will drift when $\hat{\mathrm{G}}(\mathrm{z})$ does not match the real system $\mathrm{G}(\mathrm{z})$

Another method is to make a prediction of the states, equal steps ahead for the number of steps of delay. This method is depicted schematically in Figure 4-5. With this method the output will not drift away in time. But the downside of this method is that it is sensitive to noise and disturbances and that this method will have decreased performance with higher values of k.

4-3 System Identification

As noted in Section 4-1 a model is acquired to be able to design a more sophisticated controller than a PID controller. These controllers can be for instance the control of the state space,

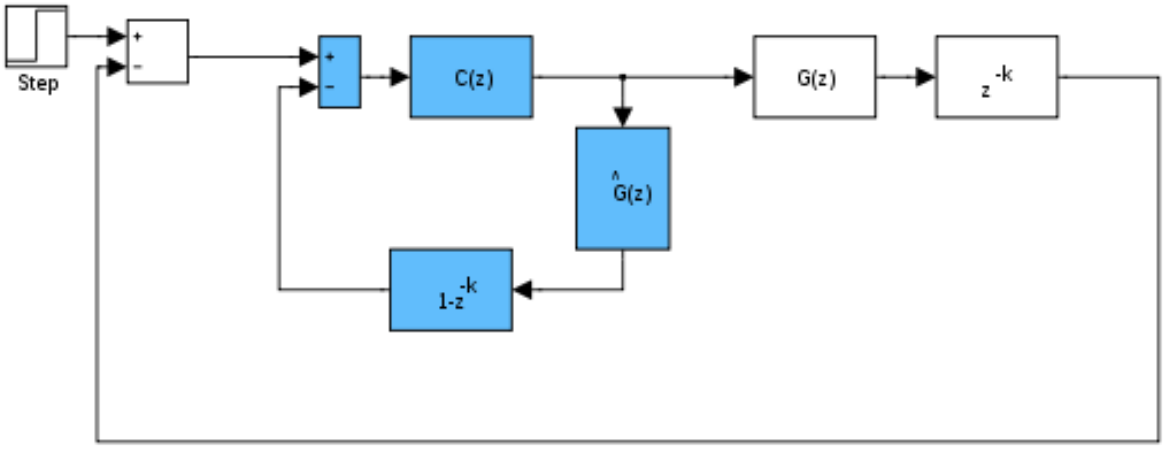


Figure 4-4: A schematic overview of the Smith Predictor

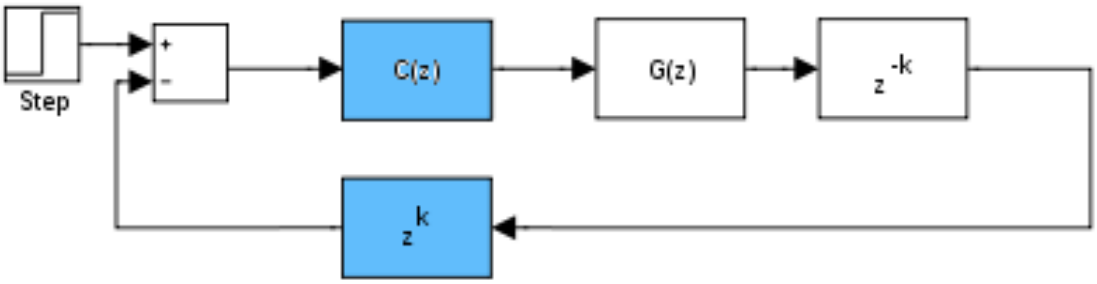


Figure 4-5: A schematic overview of the predictor with only the prediction of the output

LQG control or MPC. Besides the fact that these controllers are designed by having knowledge about the model of the system they often also rely on a model of the system for predicting future states.

There are several methods to model a system. They can be distinguished into two major categories. The first principle modelling and the experimental modelling. First principle modelling is a way of modelling in which knowledge of the behaviour of the system is put in a mathematical form. Mostly this is based on the presumed knowledge over the system and consists mostly of first principle relations like the laws of conservation of energy and mass. A well known method for this kind of modelling is the Euler Lagrange method.

This method gives a lot of insight in the system because it is built for the assumption that the system is already known. This is a problem for the UR5. It is very well possible to make a model for the robotic arm itself. The masses and kinetic movements could be measured and derived. The problem for the UR5 is that there is also an internal controller, which is unknown and also part of the system, as can be seen in Figure 4-2.

Would it be possible to know what the internal controller does, this method would be preferred, as it is a global model so it can be used for every position of the robotic arm. Therefore

it would have great value when this controller is researched more in depth or if more information about this could controller could be released.

Experimental modelling, also known as system identification, is a method of modelling which requires experimental data acquired from measurements. An input is generated and with the measured data from the outputs a model is calculated. Examples of these methods are Autoregressive model with exogenous inputs model (ARX) and Autoregressive-moving-average model with exogenous inputs model (ARMAX). These methods are based on a parametrisation of the plant and linear regression. But since not even an order of the model is known for the parametrisation of these methods, those are not chosen to identify the model of the UR5.

The method chosen for the identification of the UR5 is Subspace Identification (SID), MOESP [28]. The benefits of this method are that there is no information on the system needed beforehand. The output of this algorithm is the state space representation, instead of a transfer function. This is convenient as the targeted control method is MPC, which makes use of this representation. Furthermore this method is very generic and can be used for any other type of robotic arm.

The assumption for this method is that the system is linear. But the robotic arm is not linear. But as the internal controller of the UR5 adds gravity and friction compensation to the whole system, it is justified to regard the system as linear. Only the inertia of the robotic arm causes a non-linear behaviour. But at low speeds, these are not significant. But to make sure the best results for the control are obtained, the robot starts in a position the robot will most likely move close by during printing. This configuration is shown in Figure 4-6. Of course this starting position can be changed manually.

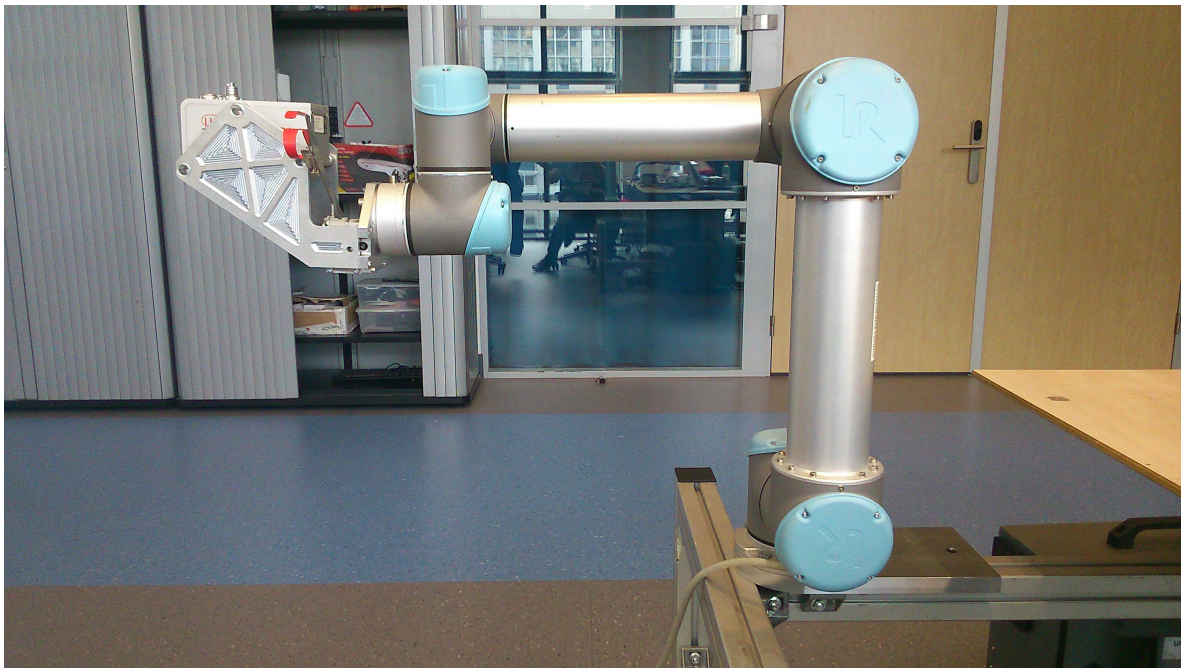


Figure 4-6: Starting configuration of the UR5 during SID

Also the identification of the robot is done for every joint individually. In other words the

robotic arm is regarded as a SISO system instead of a MIMO system. The justification for this decision is stated in Section 4-5.

The program coded for this research consists of three parts.

1. Generate the input-output data
2. Use SID to acquire the state space representation of the joints.
3. Evaluate the acquired state space representations

In the first part an input-output data is generated. Thus first an input is generated. This input sequence is sent to the UR5 to let the selected joint move. The output is retrieved after this. In this case the output is the velocity and the position of the joint. It is possible to get more information from the UR5, such as the current in the joints. This could be an interesting feature if one wants to control the force the robot exerts, as the motor current and the moment of the joint are linearly related. But for now this is not used.

Most important for the choice of the input signal for a good result for the SID is that the signal must be persistently exciting. For SID this means that the Hankel matrix [28] must not be singular. The Hankel matrix has a row dimension of $2n + 2$ where n is the order of the system to be identified. As in this thesis up to an order of 10 is evaluated, the number of sines in the multi-sine is 22. Other input signals like white noise and a block signal can be used to let the input signal be persistently exciting, but they also contain high frequencies. Typically these high frequencies are to be avoided during printing. And especially the white noise can harm the set-up.

Also it is best to perform the identification with a input signal which is close to the expected real input when the robotic arm is performing a print. It is expected that the trajectories are very smooth and low frequent and therefore it is chosen to perform the multi-sine in a band of $0Hz$ to $3Hz$.

In the second part the SID algorithm will be used. For this purpose the Predictor-Based Subspace IDentification Toolbox of Prof. M. Verhaegen will be used [29]. The inputs needed for this subroutine are the acquired data sets in step 1. Also the estimated order of the system is required. When one does not know which order this will be, the function can also show a plot of the singular values after the singular value decomposition, using the input-output data. An example of such plot is shown in Figure 4-7. Typically where the plot takes a dip, is the right choice for the order of the system. This is a compromise between the order of the model and the quality of the model. Of course it would be best if the model had infinite modes, but when using MPC one wants to keep the sizes of the state space matrices as small as possible.

In the last step the acquired model will be validated. This is done by comparing the output of the joint UR5 with the ouput of the modelled joint. The input is again a multi sine with 22 modes.

The quality of the model is tested by using the Variance Accounted For (VAF). The VAF is defined as in (4-6). In this equation y is the measured output while $\hat{y}$ is the estimated output.

$$VAF = \left(1 - \frac{var(y - \hat{y})}{var(y)}\right) \cdot 100\% \tag{4-6}$$

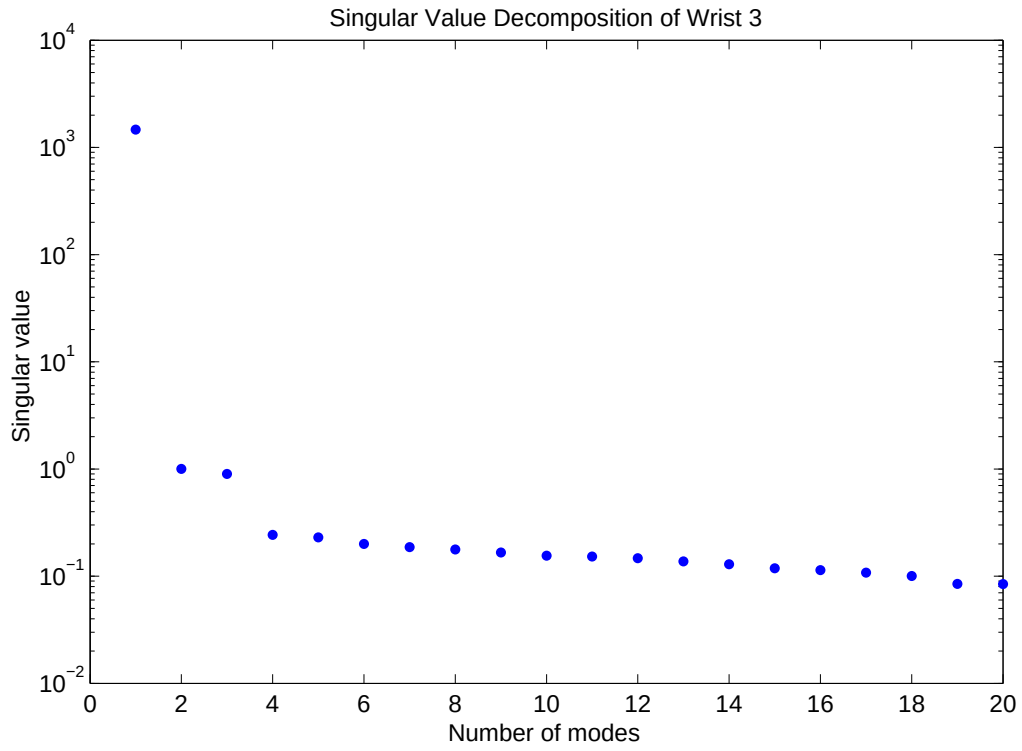


Figure 4-7: Singular value decomposition of Wrist 3 Joint

4-4 Model Validation

The results of the validation part of the identification function are listed in Table 4-1. From these results the orders of the joints are chosen. The summary of these orders is found in Table 4-2. The input reference is again a multi-sine with 22 sines between $0Hz$ and $3Hz$. The first second of the experiment is not included to filter the step response. The validation is done on the velocity of the joints, rather then the position. The reason behind this is that the position will drift in this open loop experiment. The result for the identification of the Wrist3 joint is shown in 4-8. All other plots are depicted in Appendix B.

Table 4-1: All VAF of the SID of the UR5

VAF in %	**n=2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Base Joint	79,59	99,36	99,47	99,46	99,49	99,52	99,43	99,51	99,52
Shoulder Joint	68,27	98,65	99,24	99,21	99,25	99,24	99,22	99,28	99,20
Elbow Joint	79,31	98,62	99,36	99,33	99,42	99,38	99,56	97,95	98,77
Wrist1 Joint	86,30	97,95	98,75	98,62	98,55	98,41	98,51	98,61	98,76
Wrist2 Joint	61,63	68,29	70,41	84,86	95,62	96,15	96,63	96,39	96,15
Wrist3 Joint	77,63	95,61	96,62	97,92	97,65	97,76	97,93	97,75	98,04

Table 4-2: Chosen order of the joints with corresponding VAF values

	n	VAF [%]
Base Joint	4	99.47
Shoulder Joint	4	99.24
Elbow Joint	4	99.36
Wrist1 Joint	4	98.62
Wrist2 Joint	8	96.63
Wrist3 Joint	5	97.92

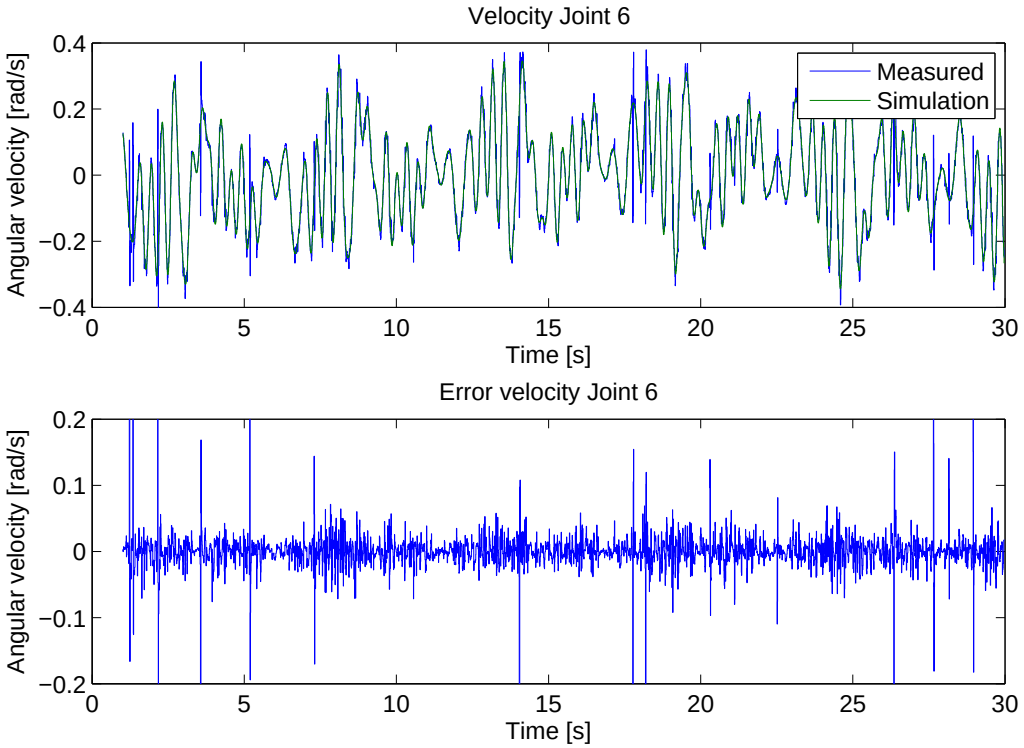


Figure 4-8: Singular value decomposition of Wrist 3 Joint

4-5 Single-Input Single-Output Approach

In this research is chosen to divide the whole system in six separate systems. The whole UR5 is simplified to six joints and it is assumed that these joints will not interfere with each other. By making this assumption the controller can be simplified. As the computational time of the MPC controller grows exponentially, six smaller systems are faster than one big system.

The cross correlation between all the joints is researched in order to determine whether it is justified to regard the system as six SISO systems. In Figure 4-9 one can see by inspecting the Bode plots that this true. In Figure 4-9a the Bode plot of the input velocity and the output velocity of the Base joint is shown. Until 10 rad/s the amplitude is 0 Decibel. This means the relation for input and output for this joint correlates. This behaviour is seen in al the six

joints. On the other hand in Figure 4-9b the Bode plot of the input velocity of the Base joint and the output velocity of the Shoulder joint is shown. Until 10 rad/s the amplitude is -30 Decibel. This means the Base joint is at lower frequencies not effected by the input velocity of the Shoulder joint. Similar behaviour is seen with all other cross correlations. Therefore it can be concluded that it is justified to have six SISO systems.

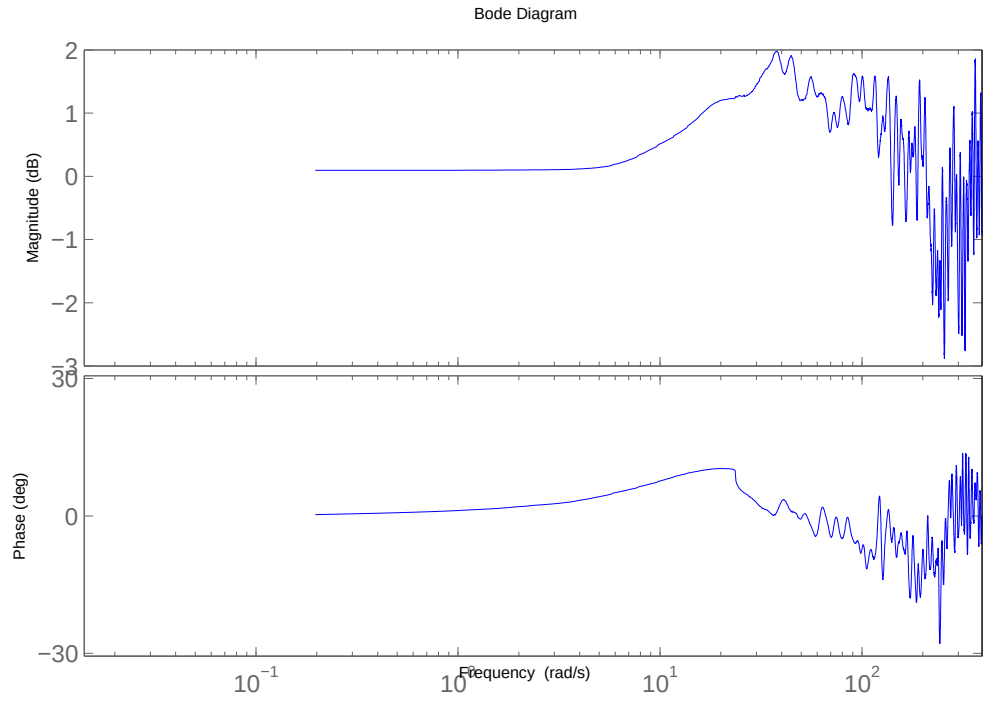
4-6 Conclusions

It is determined that there is a time-delay in the system of approximately 0.04 seconds. This corresponds to 4 time samples delay at 125 Hz. This delay can cause undesired behaviour when designing a controller. The MPC controller predicts the future 4 steps ahead to compensate for this delay.

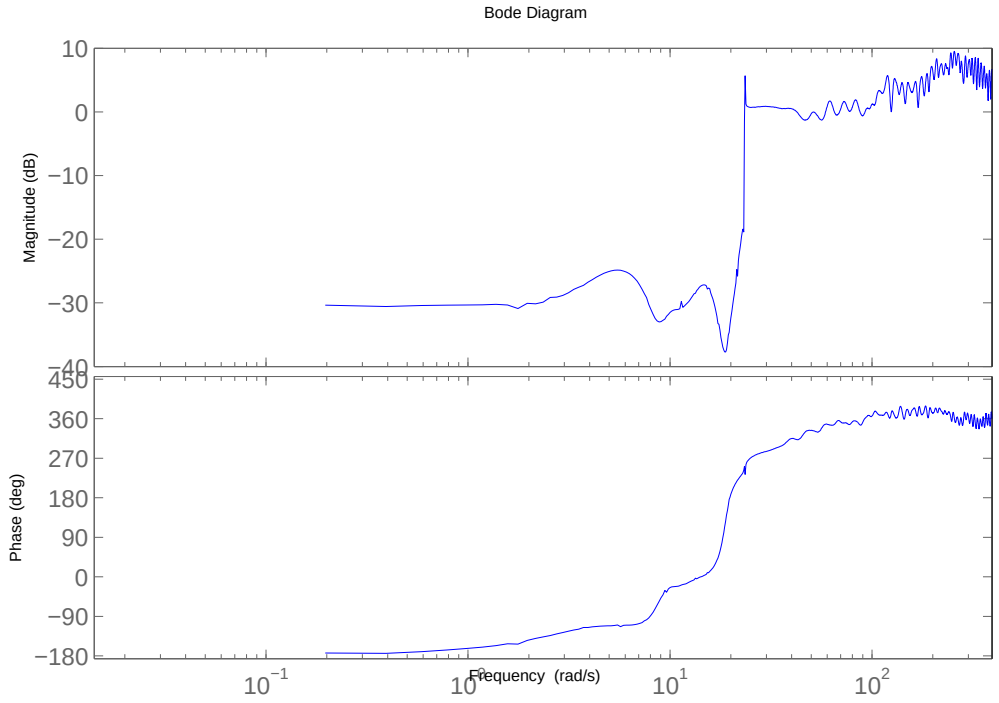
For the purpose of the MPC and the Kalman Filter a model of the UR5 is needed. It is not possible to use first principle identification methods, because a part of the whole system is unknown, namely the internal controller of the UR5. Chosen is for the SID method. This method is does not need information of the system a priori. Furthermore it is very generic and can be used in the same way for other robots. It also outputs the system in state space representation, which is very convenient as MPC can make use of this representation.

The VAF of these models lies between 95% and 100%.

It is shown that the model of the UR5 can be 6 times SISO instead of MIMO. This approach leads to smaller computational time and simplifies the whole problem.



(a) The Bode diagram of the velocity input and output for the Base joint



(b) The Bode diagram of the velocity input for the Shoulder joint and output for the Base joint

Figure 4-9: The cross correlation between the same and different joints

Chapter 5

Inverse Kinematics

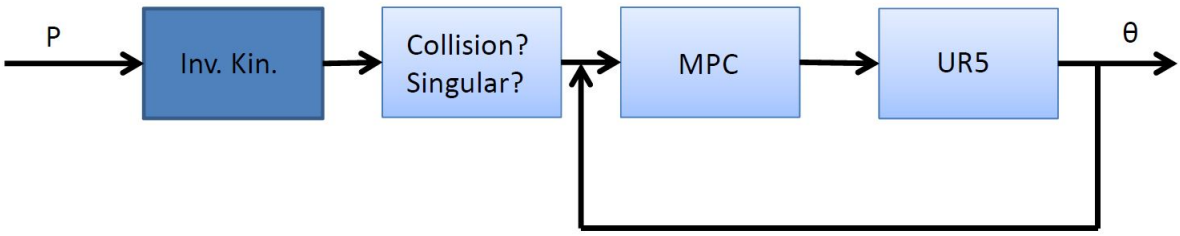


Figure 5-1: In this Chapter the Inverse Kinematics of the robotic arm will be elaborated

5-1 Inverse Kinematics Techniques

In Section 3-1 the Forward Kinematics Technique, the mapping from the joint space to the Cartesian space, for the robotic arm is discussed. In general this is an easy problem, because the transformation matrix only depends on the known joint angles and this leads to only one solution. This is illustrated in (3-3).

The mapping from the Cartesian space to the joint space is a more difficult problem for 6-DOF robotic arms. The transformation matrix also depends on the joint angles, but these are unknown in this case. Furthermore there are in general eight different configurations possible for the same Cartesian coordinates. Robotic arms like the IRB1200 from the make of Stäubli and the TX60 from the make of ABB have a spherical joint. This already makes the problem less difficult because the problem can be separated in two smaller problems. The spherical joint is solely responsible for the orientation of the TCP. The first three joints, similar to the Base, Shoulder and Elbow joint of the UR5, are responsible for the x, y and z-position of the spherical joint and through the spherical joint also the x, y and z-position of the TCP.

There are mainly two types of Inverse Kinematics Techniques, the closed form solutions and iterative techniques. The closed form solution makes use of the geometrical properties of the

robotic arm and can solve the Inverse Kinematics directly. An example of a two DOF robotic arm is given [30], to give an idea of the concept.

In Figure 5-2 a 2-link arm in shown. The lengths of link 1 and 2 are respectively l_1 and l_2. The angle from link 1 to the x-axis is expressed in θ_1. The angle of link 2 with respect to link 1 is expressed in θ_2. The position of the TCP in the Cartesian space is expressed in p. The Forward Kinematics for the position of the TCP of this robotic arm are straightforward and shown in (5-1) and (5-2). Where p_x is the x-coordinate and p_y is the y-coordinate of the TCP

$$p_x = cos(\theta_1)l_1 + cos(\theta_1 - \theta_2)l_2 \tag{5-1}$$

$$p_y = sin(\theta_1)l_1 + sin(\theta_1 - \theta_2)l_2 \tag{5-2}$$

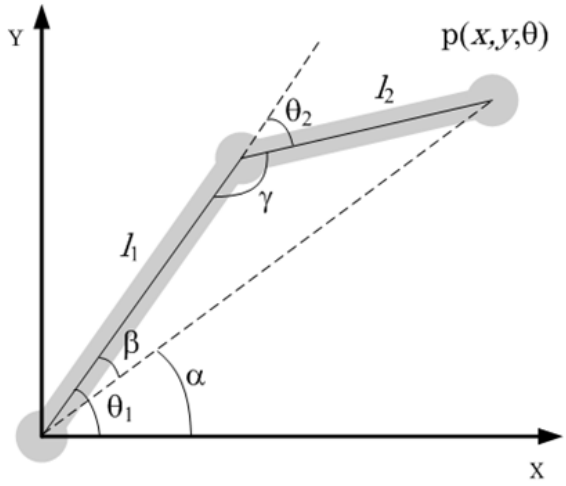


Figure 5-2: An example of Inverse Kinematics of a 2-link robotic arm

The Inverse Kinematics is more difficult. The problem now is to find the angles θ_1 and θ_2 with the known or desired position of the TCP, p.

$$(p_x^2 + p_y^2) = l_1^2 + l_2^2 + 2l_1l_2cos(\theta_2) \tag{5-3}$$

$$\theta_2 = arccos\left(\frac{(p_1^2 + p_2^2) - l_1^2 - l_2^2}{2l_1l_2}\right) \tag{5-4}$$

The angle of θ_2 can be found with the Rule of cosine in (5-3) and (5-4). Where $cos(\gamma) = -cos(\theta_2)$. Because the $arccos$ is taken, there will be two solutions for θ_2. Also $-\theta_2$ will be a solution.

$$\frac{sin(\beta)}{l_2} = \frac{sin(\theta_2)}{\sqrt{p_x^2 + p_y^2}} \tag{5-5}$$

$$\alpha = arctan\left(\frac{p_y}{p_x}\right) \tag{5-6}$$

$$\theta_1 = \alpha + \beta = arctan\left(\frac{p_y}{p_x}\right) + arcsin\left(\frac{l_2 sin(\theta_2)}{\sqrt{p_x^2 + p_y^2}}\right) \tag{5-7}$$

The angle θ_1 can be found by using the Rule of sine in (5-5) - (5-7). The angle θ_1 is dependent on θ_2, but does not have different solutions for the same θ_2. This means that there will be, in general, two different configurations for the same Cartesian position of the TCP.

For a 6-DOF robotic arm it is more difficult to find the closed form solution. With a spherical joint it is more easy to find the closed form solution, as discussed above. Unfortunately the UR5 does not have such structure. There are toolboxes available that are generating the closed form solution [31], [32]. But these toolboxes are not readily available for Matlab. Also for the UR5 and UR10 robotic arms are closed form solutions written in C++ developed by K. Hawkins [33]. So it is possible to develop this solution also in Matlab.

Another method to perform the Inverse Kinematics of the UR5 is by using iterative techniques. One of those techniques are iterative numerical techniques. It is possible to use a search algorithm, like the Newton-Raphson technique. The cost function for this optimisation algorithm can be constructed from the transformation matrix H_6^0 described in (3-3).

The twelve cost functions derived from the DH-parameters of the UR5 and described in Appendix A can be used for various minimization algorithms. This is done in the Robotics Toolbox of Peter Corke [34] with the use of the Newton-Rapson method and also various algorithms using the function $fmincon$ were tested in Matlab. Despite the fact that the initial point for the optimization was chosen closely to the desired solution, the solution of the algorithms was always stuck in a local minimum, other than desired. Therefore these methods were not used for the UR5.

The used method for the Inverse Kinematics of the UR5 is the technique where the inverse of the Jacobian is used. In the next section there will be elaborated on this method.

5-2 Inverse Jacobian Technique

While it is not possible to invert the mapping of the joint space to the Cartesian space for the position, it is possible invert it for the rates. This mapping is known as the Jacobian (3-6). This mapping from the angular velocity of the joints to the velocity of the position and orientation of the TCP is only dependent on the angles of the joints. Therefore it can be inverted to get the result of (3-1).

The strategy is to differentiate the 6D position and rotation vector generated by the printing strategy algorithm developed in [6]. The resulting vector containing the velocities of the TCP are multiplied by the inverse of the Jacobian. This Jacobian is calculated using the current angles of the joints. In Section 3-2 is explained how to calculate this Jacobian. The resulting vector represents the velocities of the joints. This vector is summed to finally result into the vector with the joint angles, the desired vector. The whole procedure is summarized in (5-8) - (5-11). Here p_i is the position and rotation vector for instant i. J_G is the geometric Jacobian and θ_i is the vector for the angles of all six joints for instance i.

$$p_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ R_{xi} \\ R_{yi} \\ R_{zi} \end{bmatrix} \tag{5-8}$$

$$\Delta p_i = p_i - p_{i-1} \tag{5-9}$$

$$\Delta \theta_i = J_G^{-1}(\theta_{i-1}) \Delta p_i \tag{5-10}$$

$$\theta_i = \theta_{i-1} + \Delta \theta_i \tag{5-11}$$

A few things should be noted for this algorithm.

- The rotation is defined as $R_i = R_{xi} R_{yi} R_{zi}$. In other words the angle is dependent on the order in which the rotation matrices are multiplied. Taking the difference is not possible without introducing an error. Only when the steps are very small this error is not significant.
- The Jacobian to calculate the next position is dependent on the current position. This will introduce an error, because θ will typically not be constant. Only when the steps are very small this error is insignificant.
- To calculate θ_i, the previous θ should be known. These angles are typically not known. The solution used in this research is to have a set of known angles with corresponding positions in Cartesian space. In this way a θ_{i-1} can be chosen which is close by the position in Cartesian space.

All the points stated above will introduce an error. Therefore the distance between the steps should be very small, in this research it will be 40 μm. In Figure 5-3 it is shown what the error will be in the resulting reference for a trajectory with a length of $1m$.

An interesting characteristic of 3D printing is that in most cases the desired trajectory is just on top of the previous trajectory. This is of course because the structure is building layer by layer. In this project the drop height is only 16 μm. This means that in contrary to the first trajectory, the steps do not have to be 40 μm or smaller. The θ_{i-1} can be taken from the previous trajectory. This is illustrated in Figure 5-4. While the bottom line needs several dots between the desired trajectory to achieve a small error, the lines above that one can directly use the data of the bottom line. In this way the algorithm will be faster.

5-3 Conclusions

The mapping from the Cartesian space to the joint space is done by the inverse of the Jacobian. The benefit of this method is that it is very generic and it can be used with any other robotic arm. But because this is a method where the velocities will be integrated to the position, there will be a small drift in the solution. But by keeping the step size small, this error also stays small, in the order of 1-10 μm at most.

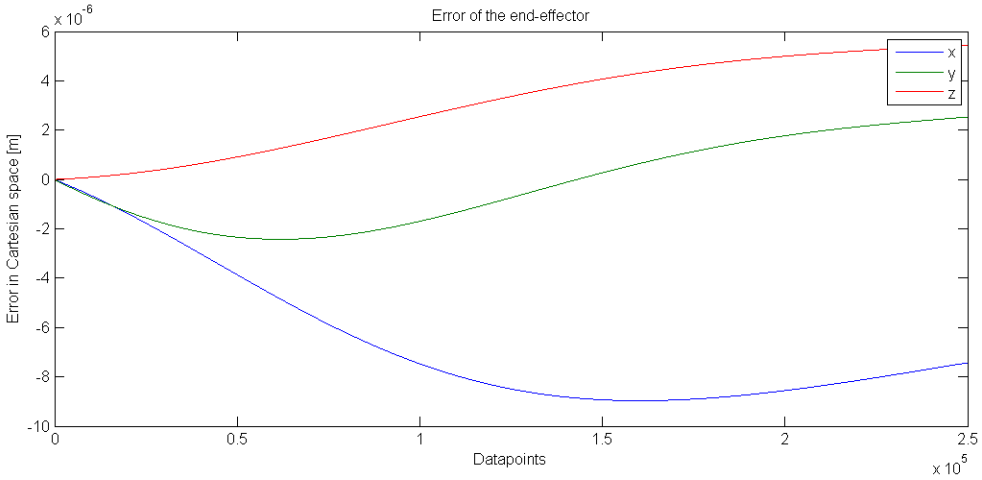


Figure 5-3: The error in x, y and z-direction for the TCP using the Inverse Jacobian method

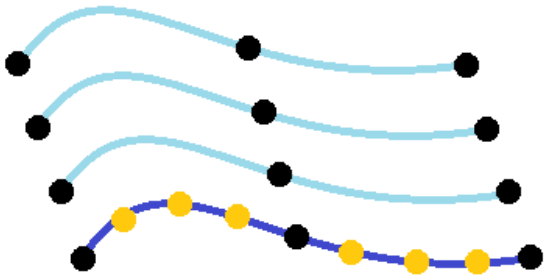


Figure 5-4: In the Inverse Jacobian algorithm previous data is used for the next layaer to speed up the algorithm

A more precise and faster method is using the closed form for the inverse kinematics. The downside of this method is that it will not be applicable to other types of robots without altering the closed form. It is also a challenge to pick the right configuration. Still this method seems to be the best, but it is not applied.

Chapter 6

Control of the UR5

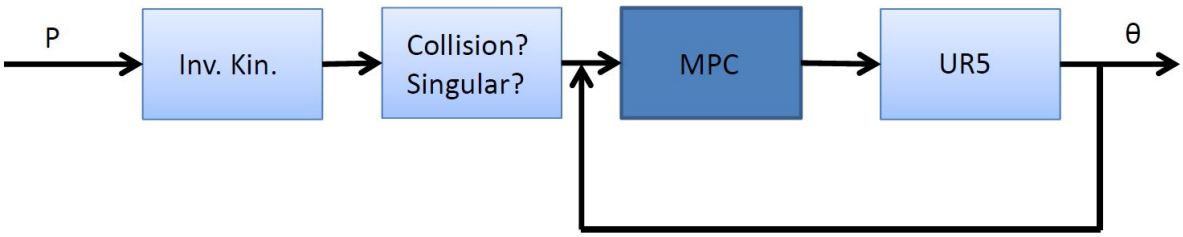


Figure 6-1: In this Chapter the control of the UR5 will be elaborated

6-1 Problem Statement

As explained in Chapter 2 the UR5 will be controlled by sending the velocity commands for the joints to the robotic arm. In this way the robotic arm is more safe. But this will give a potential problem when only relying on the internal controller of the robot. Typically a position reference for the TCP in Cartesian is given. By the Inverse Kinematics, explained in Chapter 5, this will be mapped to six references in the joint space of the individual joints. But still this will be angles and no angular velocities. The reference for the velocities can be calculated by differentiating the position reference, if the this reference is smooth. Unfortunately is this method of controlling the UR5 open loop and therefore a drift for the position will occur. When the joint deviates from his reference there is no closed loop to correct. Therefore a loop around the whole UR5 will be closed, as depicted in Figure 4-2.

6-2 Candidates for Control of the UR5

There are several methods to control the UR5. In this thesis MPC will be used to control the UR5 as a final solution. A few other methods are pole placement, LQG, PID, no additional

control at all and the method C. Kruit used in his research [5]. In this research the results of the MPC control will be compared with the situations were there is no additional control, a PID controller with feedforward and the results of the controller of J. Kruit.

How does the controller of [5] work? Making use of the laser scanner and the encoders of the robotic arm, there are five PID controllers working parallel, controlling the following properties.

- Offset distance between the printhead and the substrate (height offset controller),
- Roll angle of the TCP (roll angle controller),
- Pitch angle of the TCP (pitch angle controller),
- Yaw angle of the TCP (yaw angle controller),
- Desired printing path (direction controller).

The result is a vector Δp_i which represents the differences or rates for the position and rotation vector, as defined in (5-8). This vector is multiplied with the inverse of the Jacobian, which results in the angular velocities for the UR5. This is directly the input for the UR5. Basically the procedure for the Inverse Kinematics is used (5-9) - (5-11). Using this method it is important that the steps are small. By using this method directly in control, these steps are typically not small and therefore an error is introduced. The control of [5] typically leads to an ERMS of 1 mm.

6-3 Model Predictive Control

Model Predictive Control (MPC) is a generic term for many control algorithms which have a few key properties. According to J. Rossiter [35] these are:

- The control law depends on predicted behaviour.
- The output predictions are computed using a process model.
- The current input is determined by optimizing some measure of predicted performance.
- The receding horizon: the control input is updated at every sampling instant.

Another key property of MPC is its ability to do on-line constraint handling in a systematic way. For example to bound the input to a maximum and minimum.

In general the control problem narrows down to the optimization problem of (6-1). This problem is in general difficult to solve and optimization toolboxes are needed to solve this. But when choosing a different cost function and constraint with special properties, this problem can be made easier.

When the control problem can be defined with a quadratic cost function, H must be positive, and linear constraints (6-2), the problem can be solved by quadratic programming. Matlab

can solve these quadratic programming problems fast and therefore it is possible to perform this optimization on-line for the UR5 at a sample frequency of 125 Hz.

Finally this optimization can be even more simplified when there are no constraints. In this research there are no constraints strictly needed. The optimization will result in a quadratic equation, which means the derivative should be equalled to zero to obtain the optimal solution (6-3).

$$\begin{bmatrix} \min\limits_{x} f(x) \\ s.t.\ g(x) \leq 0 \end{bmatrix} \tag{6-1}$$

$$\begin{bmatrix} \min\limits_{x} \frac{1}{2}x^T Hx + f^T x \\ s.t.\ Ax \leq b \end{bmatrix} \tag{6-2}$$

$$\begin{gathered} \min\limits_{x} \frac{1}{2}x^T Hx + f^T x \\ Hx + f = 0 \\ x = -H^{-1}f \end{gathered} \tag{6-3}$$

The cost function in the MPC controller developed in this research has the form as in (6-4). This is only the initial form, later in this research two modifications will be done on this cost function to achieve better performance. In (6-4), J is the cost function, y_i is the output, position and velocity, of the joint at time instance i, r_i is the reference for position and velocity for the joint, u_i is the input for the joint and Q_y and Q_u are the weighting matrices for the cost function. These can be different for every time instant, but in this case they are constant. N_p is the prediction horizon. In this research the prediction horizon is defined as 30 steps.

$$J_k = \sum_{i=k}^{N_p-1+k} (y_i - r_i)^T Q_y (y_i - r_i) + u_i^T Q_u u_i \tag{6-4}$$

The challenge is now to put (6-4) in the form of (6-3). The starting point is the controllable, SISO, linear, time-invariant discrete-time state space representation of the joints, as in (6-5). The values for the A, B and C matrices are acquired by subspace identification, described in Chapter 4.

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{6-5}$$

The first step in calculating the optimal input for the system up to the horizon N_p is to calculate the predictions of $x(k)$ and $y(k)$ up to N_p. This is done by constructing the matrices as in (6-6). Or by the equivalent short notation as in (6-7). Note that in this way the predictions are only dependent on the input sequence and the states $x(k)$. In the controller these states are estimated by a Kalman filter.

$$\begin{bmatrix} x(k+1) \\ \vdots \\ x(k+N_p) \end{bmatrix} = \begin{bmatrix} A \\ \vdots \\ A^{N_p} \end{bmatrix} x(k) + \begin{bmatrix} B & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ A^{N_p-1}B & \cdots & \cdots & B \end{bmatrix} \begin{bmatrix} u(k) \\ \vdots \\ u(k+N_p-1) \end{bmatrix}$$

$$\begin{bmatrix} y(k+1) \\ \vdots \\ y(k+N_p) \end{bmatrix} = \begin{bmatrix} C & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & C \end{bmatrix} \begin{bmatrix} x(k+1) \\ \vdots \\ x(k+N_p) \end{bmatrix} \tag{6-6}$$

$$\begin{aligned} \tilde{x} &= \tilde{A}x(k) + \tilde{B}\tilde{u} \\ \tilde{y} &= \tilde{C}\tilde{x} \end{aligned} \tag{6-7}$$

By constructing $\tilde{r}$, $\tilde{Q_y}$ and $\tilde{Q_u}$ as in (6-8) similar to the construction of the other prediction matrices in (6-7), a new formulation of the cost function can be made. The cost function in (6-9) is equivalent to the cost function in (6-3). This new cost function can be rearranged to the cost function of (6-10). It should be noted that the first line of this function, $\tilde{r}^T\tilde{Q}_y\tilde{r} + x(k)^T\tilde{A}^T\tilde{C}^T\tilde{Q}_y\tilde{C}\tilde{A}x(k) - 2\tilde{r}^T\tilde{Q}_y\tilde{C}\tilde{A}x(k)$, is completely independent of the input $\tilde{u}$. Since $\tilde{u}$ is to be optimized, this term can be neglected in the optimization. The remaining terms can be arranged such that it will satisfy the minimization as in (6-3). When H and f^T are chosen as in (6-11) the optimization will return the optimal $\tilde{u}$. Only the first term is required to be fed as command to the UR5. This approach is based on [36].

$$\tilde{r} = \begin{bmatrix} r(k) \\ \vdots \\ r(k+N_p-1) \end{bmatrix}, \tilde{Q_y} = \begin{bmatrix} Q_y & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & Q_y \end{bmatrix}, \tilde{Q_u} = \begin{bmatrix} Q_u & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & Q_u \end{bmatrix} \tag{6-8}$$

$$J_k = (\tilde{y} - \tilde{r})^T\tilde{Q_y}(\tilde{y} - \tilde{r}) + \tilde{u}^T\tilde{Q_u}\tilde{u} \tag{6-9}$$

$$\begin{aligned} J_k = \tilde{r}^T\tilde{Q}_y\tilde{r} + x(k)^T\tilde{A}^T\tilde{C}^T\tilde{Q}_y\tilde{C}\tilde{A}x(k) - 2\tilde{r}^T\tilde{Q}_y\tilde{C}\tilde{A}x(k) + \\ (2x(k)^T\tilde{A}^T\tilde{C}^T\tilde{Q}_y\tilde{C}\tilde{B} - 2\tilde{r}^T\tilde{Q}_y\tilde{C}\tilde{B})\tilde{u} + \\ \tilde{u}^T(\tilde{B}^T\tilde{C}^T\tilde{Q}_y\tilde{C}\tilde{B} + \tilde{Q}_u)\tilde{u} \end{aligned} \tag{6-10}$$

$$\begin{aligned} f^T &= 2x(k)^T\tilde{A}^T\tilde{C}^T\tilde{Q}_y\tilde{C}\tilde{B} - 2\tilde{r}^T\tilde{Q}_y\tilde{C}\tilde{B} \\ H &= 2(\tilde{B}^T\tilde{C}^T\tilde{Q}_y\tilde{C}\tilde{B} + \tilde{Q}_u) \end{aligned} \tag{6-11}$$

In Section 4-2 it is reported that the UR5 has a delay of several samples. Due to this delay the performance of the controller will not be optimal. One of the proposed solutions to this problem is to predict the output several steps ahead, equal to the delay. A schematic overview is depicted in Figure 4-5. Fortunately this approach is very natural to the MPC algorithm. In (6-12) the formula is shown to compute this predicted output. In this equation k_d is the number of samples of delay. As long as the delay is smaller than the prediction horizon of

the MPC, the values for $u(k-i-1)$ can be found in $\tilde{u}$. Finally $x(k)$ should be replaced by the new value $x_{delay}(k)$ to obtain a better performing MPC.

$$x_{delay}(k) = x(k+k_d) = A^{k_d}x(k) + \sum_{i=0}^{k_d-1} A^i B u(k-i-1) \tag{6-12}$$

In equation (6-4) it can be seen that the minimization is dependent on $(y-r)$ and u. For the robotic arm those terms can typically not both be zero, as the UR5 tries to track a reference in position. This means a velocity input is sent to the UR5. This means that depending on the weighting matrices the error and the input will be balanced. Unless Q_u is not zero, $(y-r)$ will be non-zero. Equalling Q_u to zero is not a good strategy, as the input will become very jittery.

A better approach is to make use of an Increment-Input-Output (IIO) model [37]. By doing this in this research the new cost function will be of the form as shown in (6-13). Instead of the input, the increment of the input is now in this function. Still Δu will typically not be zero, when the error between the output and the reference is zero. But because in this research the accelerations of the joint are not big, this approach gives a better result. This is shown in Figure 6-5 in the next Section.

This IIO approach is easily applicable to the MPC architecture. The increment of the input is defined as in (6-14). Using this definition a new state can be introduced to the statevector, namely the input, as in (6-15). Finally the state space matrices acquired by the SID can be augmented with these equations to get the result of (6-16). These augmented matrices can be used for the MPC in the same way as the normal matrices as described above.

$$J_k = \sum_{i=k}^{N_p-1+k} (y_i - r_i)^T W_y (y_i - r_i) + \Delta u_i^T W_u \Delta u_i \tag{6-13}$$

$$u(k) = u(k-1) + \Delta u(k) \tag{6-14}$$

$$x_u(k) = u(k-1) \tag{6-15}$$

$$\begin{aligned} \begin{bmatrix} x(k+1) \\ x_u(k+1) \end{bmatrix} &= \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ x_u(k) \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \Delta u(k) \\ y(k) &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ x_u(k) \end{bmatrix} \end{aligned} \tag{6-16}$$

6-4 Experimental Results

In this Section four different tests are performed to compare different situations. The experiments are:

- Test the MPC for different speeds

- Test the MPC for different directions
- Compare the IO model and the IIO model
- Compare different kinds of control

The plots in this Chapter are the errors for the print, thus after calculations. The plots for the individual joints can be found in Appendix C.

The most important results of this Chapter are the the accuracy and the repeatability. In the requirements in Section 1-4 it is stated that these values should be under 100 μm.

In the results of this Section the error is defined as the difference in distance between the position where the drop of ink should hit on the surface and the position where the drop of ink will hit. The assumption is made that the object is flat and parallel to the print head. The results are calculated using the angles of the encoders of the UR5.

The accuracy is defined as an Error Root-Mean-Square (ERMS) as in (6-17). In this equation N is the number of samples, r is the reference in position for the print and y is the measured position for the print.

The repeatability is defined as a standard deviation in the x and y direction of the print. This is shown in (6-18). In this equation is M the number of experiments, y is the position of the print in x or y-direction an μ is the mean of the experiments for a sample.

$$ERMS = \sqrt{\frac{1}{N}\sum_{N}^{i=1}(y_i - r_i)^2} \tag{6-17}$$

$$\sigma_x = \frac{1}{N}\sum_{N}^{j=1}\left(\sqrt{\frac{1}{M}\sum_{M}^{i=1}(y_{xi} - \mu_{xi})^2}\right)_j \tag{6-18}$$

$$\sigma = \sqrt{\sigma_x^2 + \sigma_y^2} \tag{6-19}$$

6-4-1 Different velocities

In this test the same trajectory will be tracked at different speeds. The interesting part of this experiment is the trade-off between the velocity of the TCP on the one hand and the accuracy and repeatability on the other hand. Intuitively one should say that a higher velocity would lead to a decrease of performance.

The trajectory is a straight line in the x-direction, according to Figure 6-2, for 200 mm. At the same time the print head will turn around the x-axis for 0.125π. This is done for four different speeds, $2.5mm/s$, $5mm/s$, $10mm/s$ and $40mm/s$.

The absolute error for this experiment is plotted in Figure 6-3. The ERMS and repeatability for the different velocities are shown in Table 6-1. For the calculation of the repeatability are ten experiments conducted.

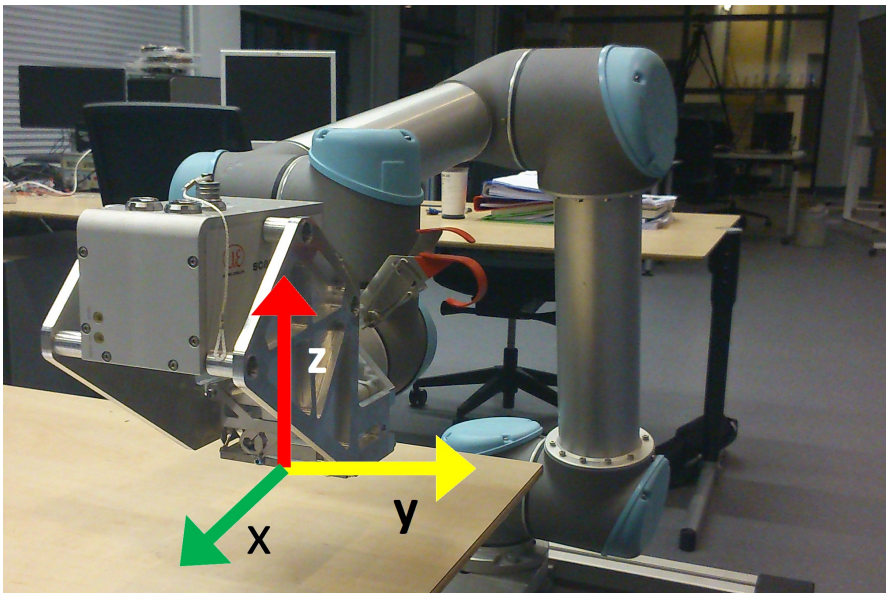


Figure 6-2: Picture of the directions of the trajectory

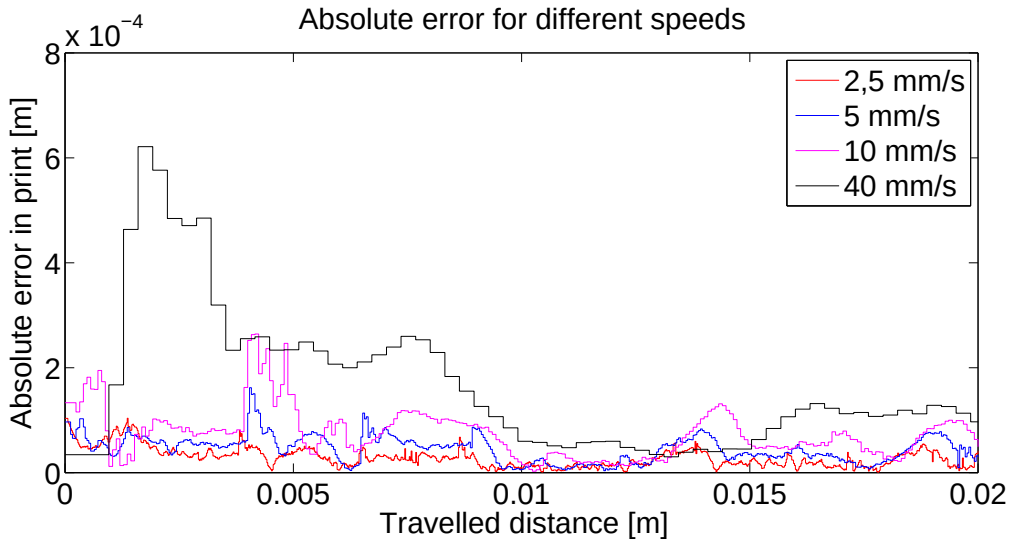


Figure 6-3: Plot of the absolute error of the print for different velocities of the TCP

The conclusion to be drawn from this experiment is that the performance in accuracy and repeatability decreases when the velocity increases. Although it is not directly proportional. In can be an interesting problem to find the maximum velocity given a maximum allowable error and repeatability.

6-4-2 Different directions

In this experiment the TCP will track a trajectory which is $200mm$ in respectively x, y and z direction, according to Figure 6-2. In all these trajectories the TCP will also rotate along

Table 6-1: The ERMS values of the print for different velocities of the TCP

	2.5 mm/s	5 mm/s	10 mm/s	40 mm/s
ERMS [μm]	27.9	48.4	69.3	130.5
σ_x [μm]	11.3	14.9	31.3	74.8
σ_y [μm]	9.4	10.0	9.6	13.5
σ [μm]	14.7	17.9	32.8	76.0

the x-axis for 0.125π. The velocity of the TCP is $10mm/s$. The results are shown in Figure 6-4 and Table 6-5.

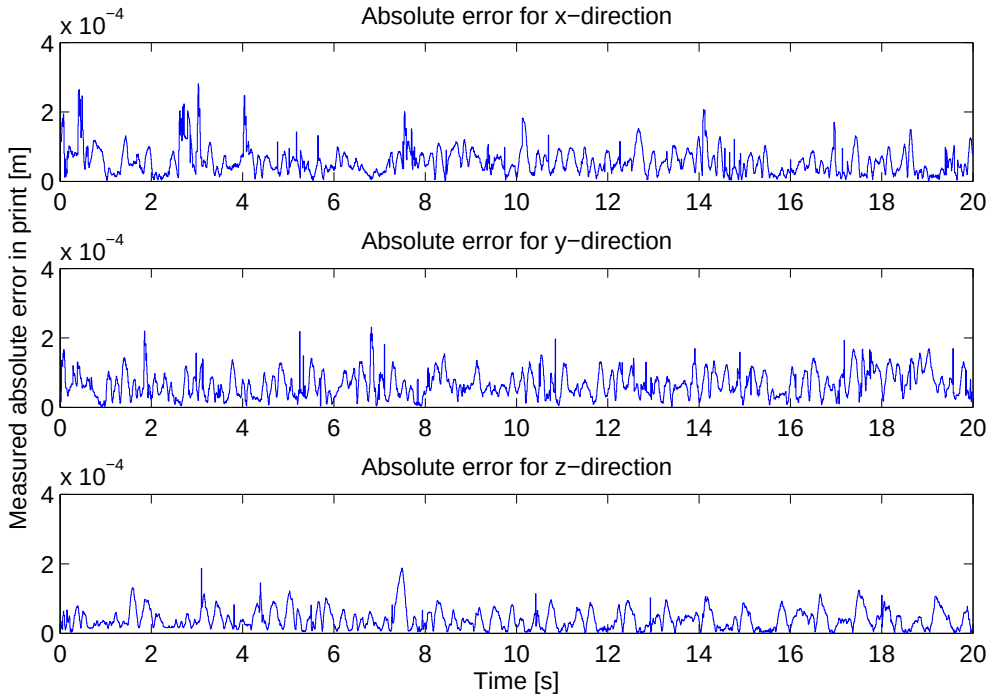


Figure 6-4: Plot of the absolute error of the print for different directions of movement of the TCP

The conclusion drawn from this experiment is that the error will be at largest in the direction of movement. Of course this will mean that a trajectory in z-direction will give the smallest error.

6-4-3 IIO model and IO model

In this experiment the TCP will track a trajectory which is $200mm$ in x direction, according to Figure 6-2. In the trajectories the TCP will also rotate along the x-axis for 0.125π. The velocity of the TCP is $10mm/s$. The difference between the experiments is that the one is run with a Increment-Input-Output (IIO) model and the other with a Input-Output (IO) model. Details on this models can be found in Section 6-3.

Table 6-2: The ERMS values of the print for different directions of movement of the TCP

	x-direction	y-direction	z-direction
ERMS [μm]	69.3	76.8	49.1
σ_x [μm]	31.3	4.9	5.6
σ_y [μm]	9.6	19.9	2.2
σ [μm]	32.8	20.5	6.0

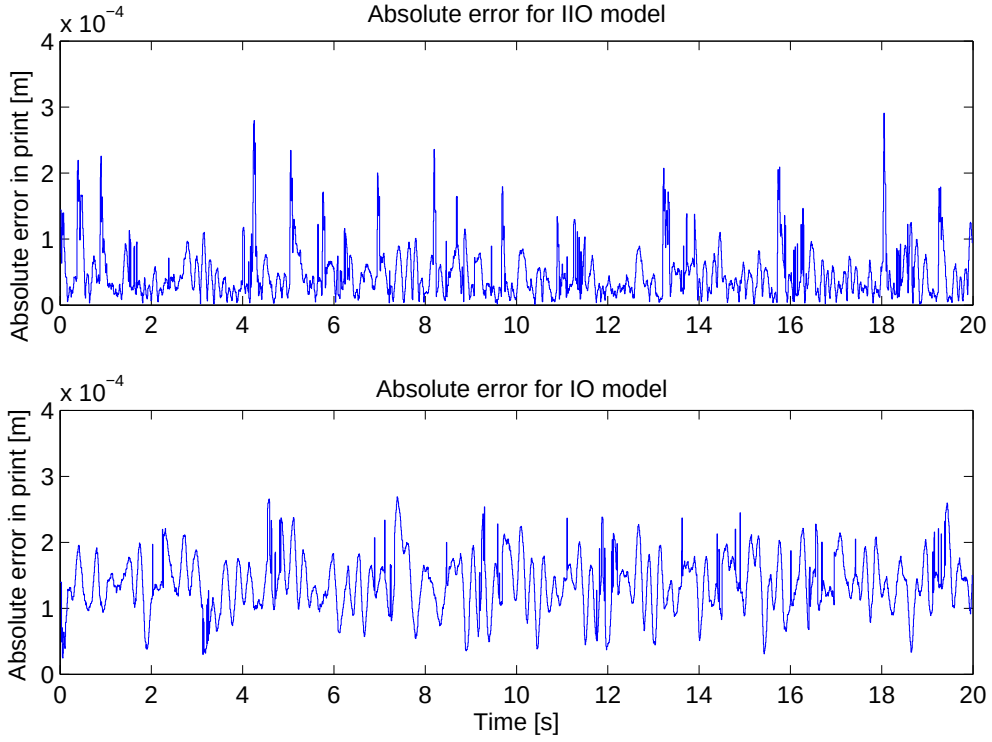


Figure 6-5: Plot of the absolute error of the print for IIO and IO models

The conclusion from this result is that the accuracy and the repeatability of the print is better when using the IIO model. The IIO model is lagging less behind the IO model.

6-4-4 Different control methods

In this experiment the TCP will track a trajectory which is $200mm$ in x direction, according to Figure 6-2. In the trajectories the TCP will also rotate along the x-axis for 0.125π. The velocity of the TCP is $10mm/s$. The difference between the experiments is that the first is completed with a Model Predictive Control (MPC) controller, the second with a PID controller with a feed forward term according to (6-20). Here r_v is the reference for the velocity of the joint, y_v is the measured velocity and K is the gain for the PID controller. The last experiment is performed without additional controller, which means its input is the velocity reference.

Table 6-3: The ERMS values of the print for IIO and IO models

	IIO	IO
ERMS [μm]	69.3	144.2
σ_x [μm]	31.3	39.1
σ_y [μm]	9.6	9.5
σ [μm]	32.8	40.2

$$u(k) = r_v(k) + K(r_v(k) - y_v(k-1)) \tag{6-20}$$

The results of the experiments are shown in Figure 6-6 and Table 6-4.

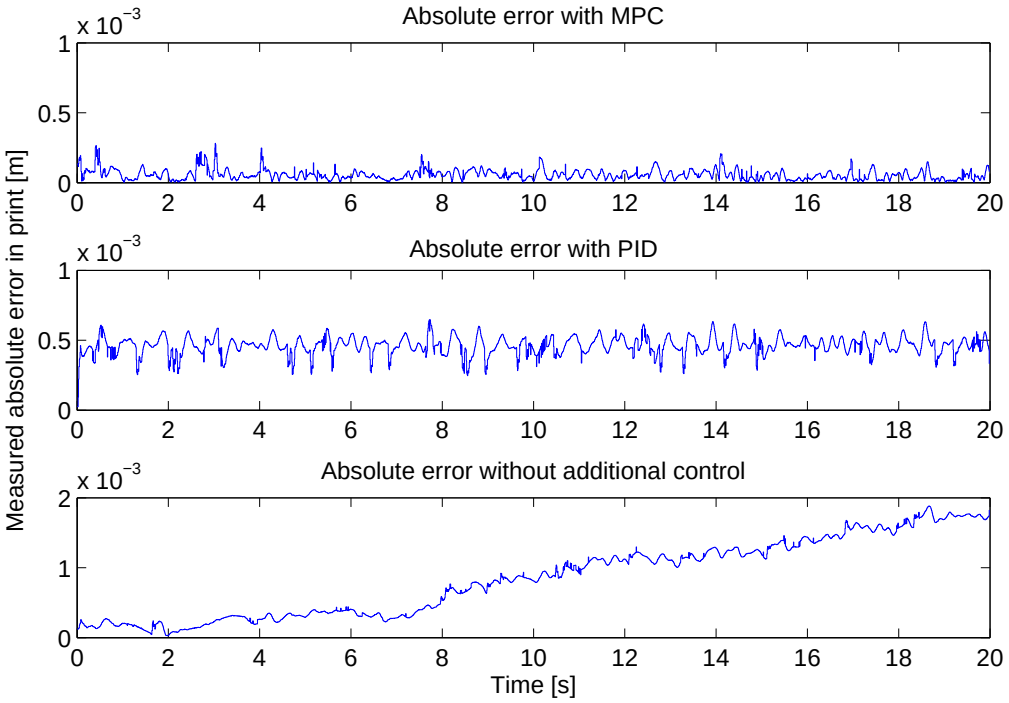


Figure 6-6: Plot of the absolute error of the print for different control techniques

According to the experiments, the MPC controller outperforms the PID and the case without an additional controller in term of accuracy. The PID is outperforming the MPC controller on repeatability. On this moment there is no good explanation why this is the case. As can be seen in Figure 6-6 the PID controller does have a worse performance on the variance on the absolute error. Also compare to the controller designed in the research of C. Kruit a worse performance with an accuracy of $1mm$ was achieved [5]. The MPC controller is an improvement on that.

Table 6-4: The ERMS values of the print for different control techniques

	MPC	PID	No control
ERMS [μm]	69.3	459	1100
σ_x [μm]	31.3	19.1	394.8
σ_y [μm]	9.6	5.8	24.4
σ [μm]	32.8	20.0	395.5

6-5 Conclusions

The MPC controller is used without constraints. Therefore the cost function has a global minimum and no search algorithm is needed for finding this minimum. The MPC controller has a horizon of 30 steps. Also a Kalman filter is in this controller to predict the states. The time-delay of the system is taken care of by this controller as it predicts the next 4 steps for the states. With altering the structure of the cost function, such that the input becomes part of the states and the difference of the sequential inputs becomes the new input, so using a IIO model, better results are achieved.

At 10 mm/s the MPC controller achieves an ERMS of 69.3 μm and a repeatability of 32.8 μm. For the same trajectory the PID-controller achieves an ERMS of 459 μm and a repeatability of 20.0 μm. The MPC controller als outperforms the case where there is no additional control. In this experiment the print had an ERMS of 1100 μm and a repeatability of 395.5 μm. The control in the research of C. Kruit achieved an accuracy of approximately 1000 μm [5].

Chapter 7

Conclusions

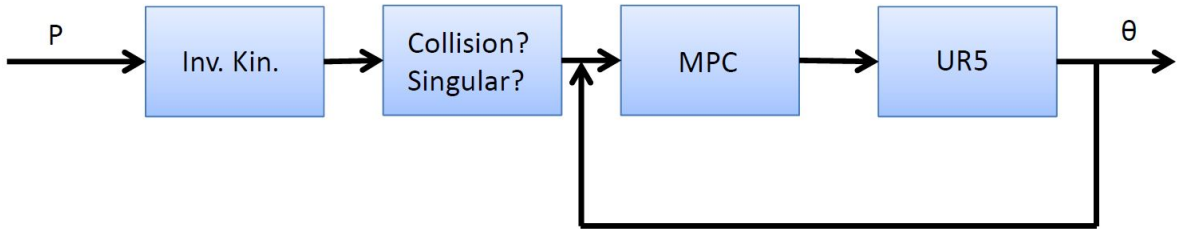


Figure 7-1: Overview of this research

7-1 Overall system

There are a few methods to command the UR5. The main methods are via ROS using the URControl, via Matlab using the UR control and via C-API. The chosen method is via Matlab as all other programs are also developed in Matlab and this ensures fast communication between the programs. Within the Matlab Driver no Simulink is used and the velocities in the joint space of the robot are commanded. As this gives the best performance and is safest. The robotic arm can be controlled at a maximum sampling frequency of 125 Hz.

Using the Denavit-Hartenberg parameters a kinematic model could be derived of the UR5. This model is extended with the print head and the laser scanner.

When the reference in the joint space is calculated it will be checked whether is lies in the workspace of the UR5. By checking if the velocity of the joint does not exceed the maximum velocity of the joint, it will be prevented that the UR5 will suffer from near-singular configurations. The solution to this is to slow down the movement of the robot in this part of the workspace. Or by finding an alternative trajectory for the print, as suggested in Section 8-3.

It is also checked whether the robotic arm will collide into itself. At the moment this is done by restricting the joints to exceed predetermined angles. This is a very conservative method and it is certain that workspace of the UR5 where no collision takes place is restricted by this method. A proposed method is the Separating Axis Theorem (SAT) method. When the UR5, the environment and the print will be enveloped by Oriented Bounding Box (OBB)'s, SAT proves to be very fast and efficient.

7-2 Modelling

It is determined that there is a time-delay in the system of approximately 0.04 seconds. This corresponds to 4 time samples delay at 125 Hz. This delay can cause undesired behaviour when designing a controller. The Model Predictive Control (MPC) controller predicts the future 4 steps ahead to compensate for this delay.

For the purpose of the MPC and the Kalman Filter a model of the UR5 is needed. It is not possible to use first principle identification methods, because a part of the whole system is unknown, namely the internal controller of the UR5. Chosen is for the Subspace Identification (SID) method. This method does not need information of the system a priori. Furthermore it is very generic and can be used in the same way for other robots. It also outputs the system in state space representation, which is very convenient as MPC can make use of this representation.

The Variance Accounted For (VAF) of these models lies between 95% and 100%.

It is shown that the model of the UR5 can be 6 times SISO instead of MIMO. This approach leads to smaller computational time and simplifies the whole problem.

7-3 Inverse Kinematics

The mapping from the Cartesian space to the joint space is done by the inverse of the Jacobian. The benefit of this method is that it is very generic and it can be used with any other robotic arm. But because this is a method where the velocities will be integrated to the position, there will be a small drift in the solution. But by keeping the step size small, this error also stays small, in the order of 1-10 μm at most.

A more precise and faster method is using the closed form for the inverse kinematics. The downside of this method is that it will not be applicable to other types of robots without altering the closed form. It is also a challenge to pick the right configuration. Still this method seems to be the best, but it is not applied because of lack of time to fully test the algorithm. This is also explained in Section 8-1.

7-4 Control

The MPC controller is used without constraints. Therefore the cost function has a global minimum and no search algorithm is needed for finding this minimum. The MPC controller has a horizon of 30 steps. Also a Kalman filter is in this controller to predict the states. The

time-delay of the system is taken care of by this controller as it predicts the next 4 steps for the states. With altering the structure of the cost function, such that the input becomes part of the states and the difference of the sequential inputs becomes the new input, so using a Increment-Input-Output (IIO) model, better results are achieved.

At 10 mm/s the MPC controller achieves an ERMS of 69.3 μm and a repeatability of 32.8 μm. For the same trajectory the PID-controller achieves an ERMS of 459 μm and a repeatability of 20.0 μm. The MPC controller als outperforms the case where there is no additional control. In this experiment the print had an Error Root-Mean-Square (ERMS) of 1100 μm and a repeatability of 395.5 μm. The control in the research of C. Kruit achieved an accuracy of approximately 1000 μm [5].

7-5 Goals and Requirement

- **Explore the possibilities for the UR5.**

  Research has been conducted about the best way of controlling the UR5. By controlling the velocity in joints space it can perform control tasks.

- **Achieve safe control of the UR5.**

  Several measures are taken to make the UR5 safer. Coro [14] has shown that the force of the UR5 should not be underestimated. The safety measures consist of a check whether the robot is still moving as predicted. If this is not the case, the robot shuts down. Also the decision to only command the joints of the UR5 is decided from a safety perspective.

- **Develop a method to avoid near-singular configurations and collision with itself and other objects.**

  The rates of the joints are checked for near singularities and the angles of the joints are restricted such that the UR5 cannot collide to itself.

- **Develop a dynamic model for the UR5.**

  Using SID a dynamic model could be retrieved with a VAF between 95% and 100%.

- **The controller should cope with as input a trajectory in Cartesian space added with a rotational vector ($[x, y, z, Rx, Ry, Rz]$).**

  The developed system can handle the rotational vector as stated above. This vector is the input to the inverse kinematics program. The output is a six dimensional vector in joint space.

- **Achieve a precision of 100 μm for the print.**

  For a velocity of 10 mm/s of the print, the ERMS is 69 μm on the print, when using the MPC approach.

- **All operation should be executable from one device.**

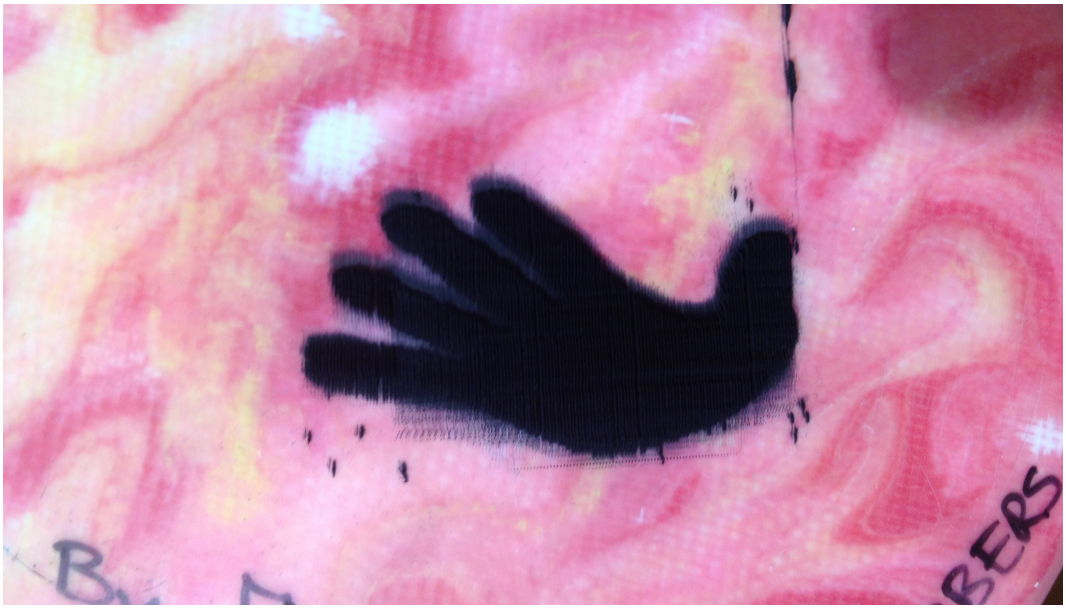
  The system is operational on one 32bit pc with Windows. Research is still going on making the system operational on a 64bit pc.

7-6 Experimental printing result

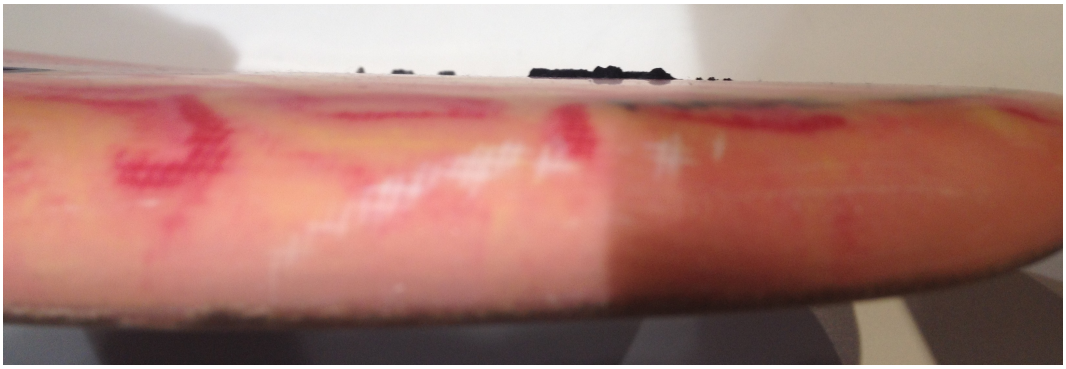
During the fair Dig-It in Delft in November 2014, the whole set-up has successfully printed a structure on a double curved surface. The goal was to print a small hand on a hand surf board. During the day approximately 200 layers have been printed to result in the print shown in Figure 7-2. The height of the print was finally between 1 and 2 mm.

Two observations can be made from this print. The layer were very neatly stacked on top of each other. Only in the printing direction the error was in some prints very large. This can be explained because the printer sometimes stalled. This is also explained in Section 8-2.

Furthermore the lines in the print are not as straight as they should be. A small oscillation is noticeable. Because this oscillation is repeated for around 200 experiments the assumption is that this oscillation is not caused by noise but something structural. A possible reason could be that the parameters of the robotic arm are not correct. This is also explained in Section 8-4



(a) The print of the hand



(b) A view of the height of the print

Figure 7-2: 3D print of a small hand on a small surfboard

Chapter 8

Future Work

In this research results are achieved and conclusions are drawn from that. These results and conclusions can be read in Chapter 7. From this research interesting questions and problems appeared. These questions and problems are not tackled because they were out of the scope of this project or due to lack of time.

8-1 Inverse Kinamatics

In Chapter 5 can be read that for this research a method of Inverse Kinematics had to be chosen. It is chosen in this research to use the method with the inverse Jacobian. This method is very generic and fits in the requirements of this project.

An other method mentioned is to make use of the closed form solution. Although this method is not generic for all kind of robotic arms, it can be developed for the UR5. In fact it is explained in [33] how to do so. Also a program has been developed for this study and it proves to be a very fast method and will provide all the possible solutions. But this program is not yet tested fully and therefore not yet safe to use. How must be decided which of the solutions have to be chosen, for example.

Because this method is very fast it can be used to do on-line calculations. This can be interesting when the controller should act on the Cartesian space, instead on the joint space as in this research.

8-2 Upgrading to a 64-bit pc

The whole set-up, so including the part in [6], is only working currently on a 32-bit device. This is mainly because the driver for the print head is only working with a 32-bit driver. Also the MEX-file for the laser scanner is not available for a 64-bit pc. The driver and the controller for the UR5 do work on a 64-bit pc. But the robotic arm is not performing perfect

for the 32-bit set-up. During a run the connection from the pc to the robot seems to be lacking. This causes the robot to stop in the middle of a print and this will cause major errors for the print in the direction of printing.

A solution to this is to keep the whole set-up in a 32-bit environment and to find the solution to the bad performance of the UR5 described above.

The other solution is to transfer the whole set-up to a 64-bit environment, but then there must be a solution to the drivers for the scanner and the print head.

8-3 Alter Starting Point

In Section 3-2 and 3-3 it is explained how there can be detected by the system if there will be a collision or singularity in the current defined print path. It can even be determined if the system will suffer from near-singularities.

Unfortunately the system can only detect whether the print path is infeasible or undesired and give a notification to the user. It would be a better solution if the system could determine what the starting point of the print should be for an optimal trajectory and therefore minimizing the error of the print.

The difficulty for this problem is that the trajectory will be defined in Cartesian space while the restrictions are defined in the joint space. It is possible to define 8 different trajectories in joint space with Inverse Kinematics, but still it will be difficult to move this trajectory due to the high non-linear structure of the robotic arm.

8-4 Research Mechanical Inaccuracies

The results in this research are based upon the data the robotic arm returns. In other words the assumption is made that the kinematic model of the UR5 is perfect. Most probably this is not the case, which means the performance of the robotic arm will be somewhat worse than the results reported in this research.

It would improve the results on the print if these accuracies would be known or if a calibration routine could be run on the UR5. For this purpose the laser scanner already attached to the robotic arm to scan known objects. Comparing the scanned results to the known dimensions of the object can calibrate the robotic arm.

Appendix A

Transformation Matrix H_6^0

The equations in this Appendix form the transformation matrix for the Forward Kinematics of the UR5, as defined in Chapter 2. In these equations $s_{i+j} = sin(\theta_i + \theta_j)$ and $c_{i+j} = cos(\theta_i + \theta_j)$. Furthermore the Denavit-Hartenberg (DH) coefficients for d_i and a_i can be found in Figure 3-3.

These equations can be used in the cost function in an optimization algorithm to find the Inverse Kinematics. This is explained in Chapter 5.

$$H_6^0 = \begin{bmatrix} R_{11} & R_{12} & R_{13} & x \\ R_{21} & R_{22} & R_{23} & y \\ R_{31} & R_{32} & R_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A-1}$$

$$\begin{aligned}
R_{11} &= c_6(s_1 s_5 + c_{234} c_1 c_5) - s_{234} c_1 s_6 \\
R_{21} &= -c_6(c_1 s_5 - c_{234} c_5 s_1) - s_{234} s_1 s_6 \\
R_{31} &= c_{234} s_6 + s_{234} c_5 c_6 \\
\\
R_{12} &= -s_6(s_1 s_5 + c_{234} c_1 c_5) - s_{234} c_1 c_6 \\
R_{22} &= s_6(c_1 s_5 - c_{234} c_5 s_1) - s_{234} c_6 s_1 \\
R_{32} &= c_{234} c_6 + s_{234} c_5 s_6 \\
\\
R_{13} &= c_5 s_1 - c_{234} c_1 s_5 \\
R_{23} &= -c_1 c_5 - c_{234} s_1 s_5 \\
R_{33} &= -s_{234} s_5
\end{aligned} \tag{A-2}$$

$$\begin{aligned}
x &= d_6(c_5 s_1 - c_{234} c_1 s_5) + d_4 s_1 + c_1(a_3 c_{23} + a_2 c_2) + d_5 s_{234} c_1 \\
y &= s_1(a_3 c_{23} + a_2 c_2) - d_4 c_1 - d_6(c_1 c_5 + c_{234} s_1 s_5) + d_5 s_{234} s_1 \\
z &= d_1 + a_3 s_{23} + a_2 s_2 - d_5 c_{234} - d_6 s_{234} s_5
\end{aligned}$$

Appendix B

Validation Plots

In this Appendix all the plots for the validation of the joints of the UR5 are shown. The validation input is a multi-sine with 22 sines. The frequency of those sines is between $0Hz$ and $3Hz$. Because the input is the reference for the velocity for the UR5, the output depicted resembles this input.

Only the velocity is relevant for the validation. The position is not relevant, because this will have a drift. This is because the validation is open loop. The first second of the validation is not used, to get rid of undesired high-frequent dynamics.

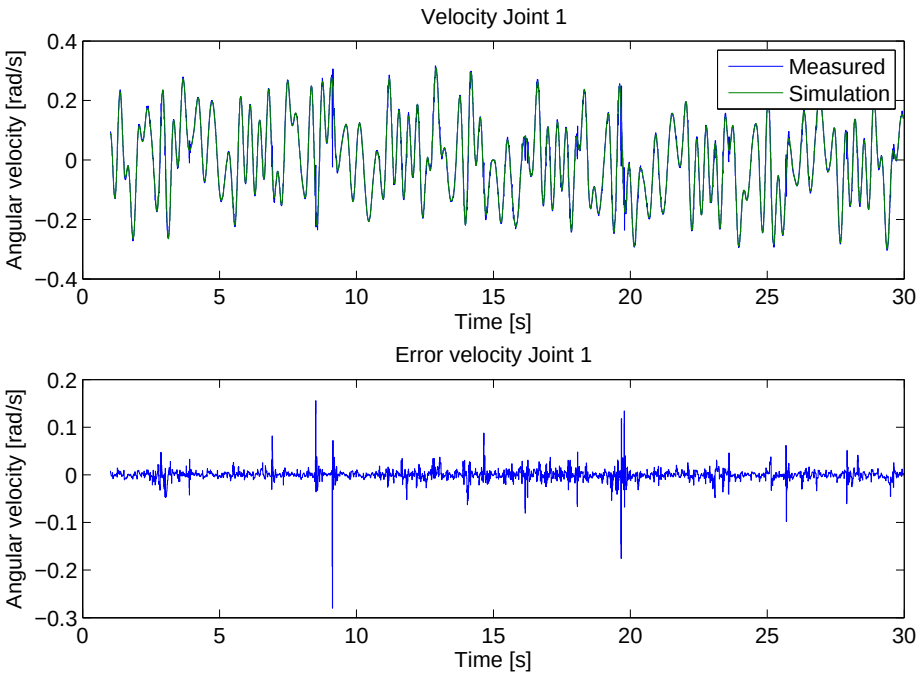


Figure B-1: Validation results for the Base Joint

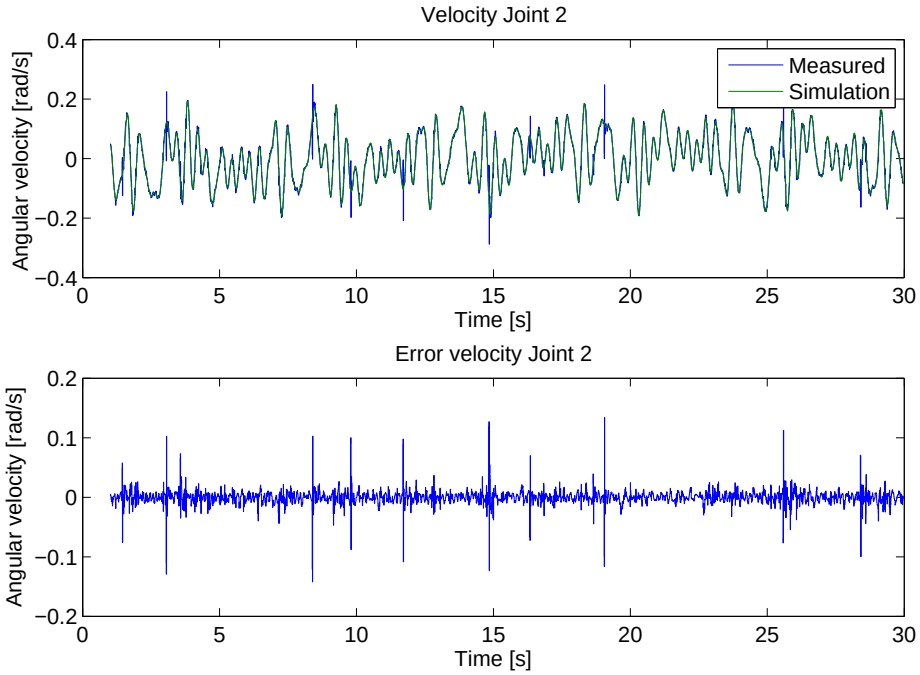


Figure B-2: Validation results for the Shoulder Joint

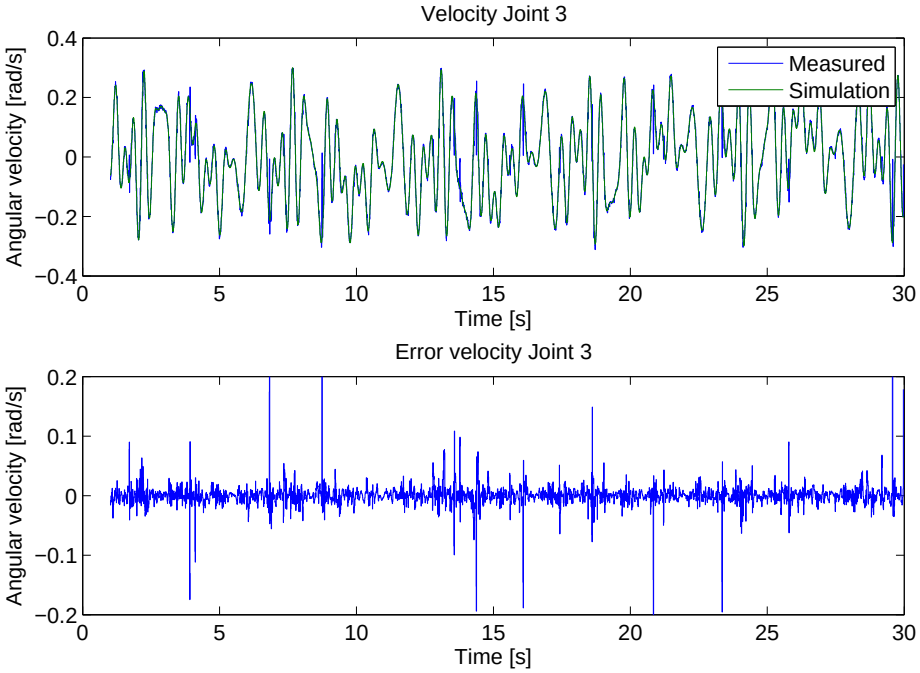


Figure B-3: Validation results for the Elbow Joint

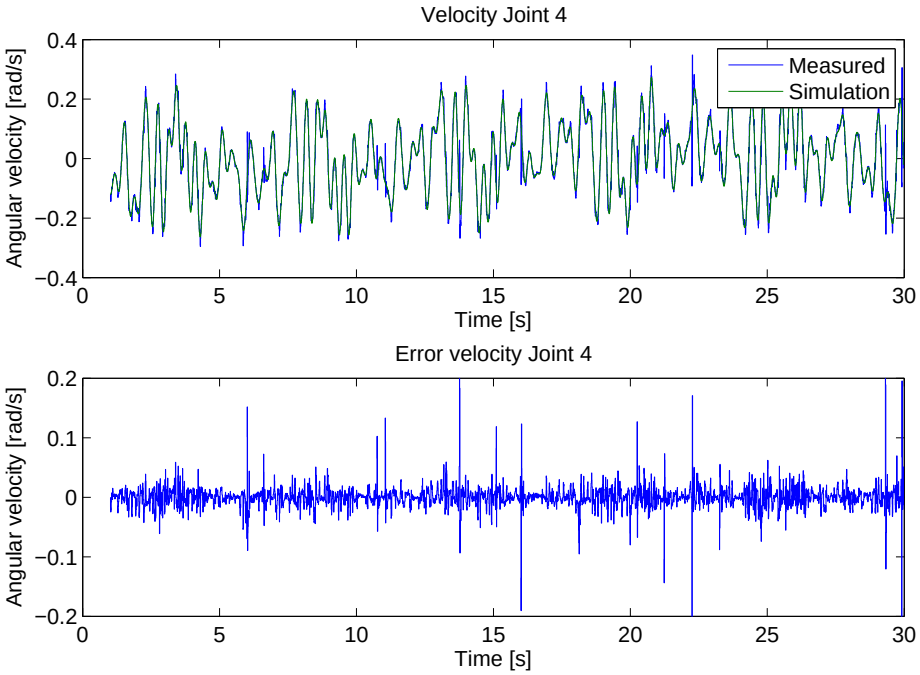


Figure B-4: Validation results for the Wrist1 Joint

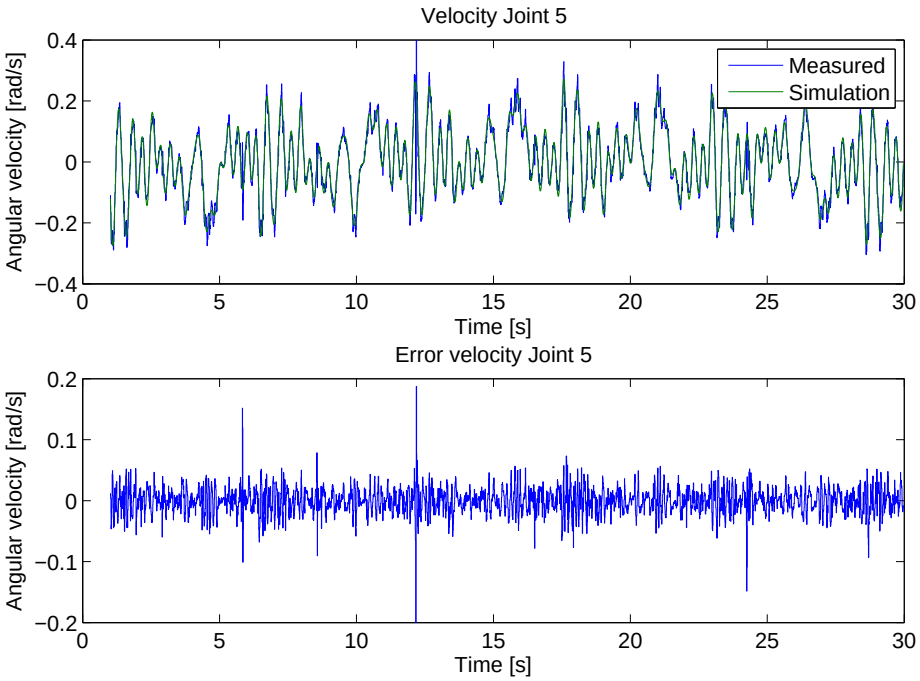


Figure B-5: Validation results for the Wrist2 Joint

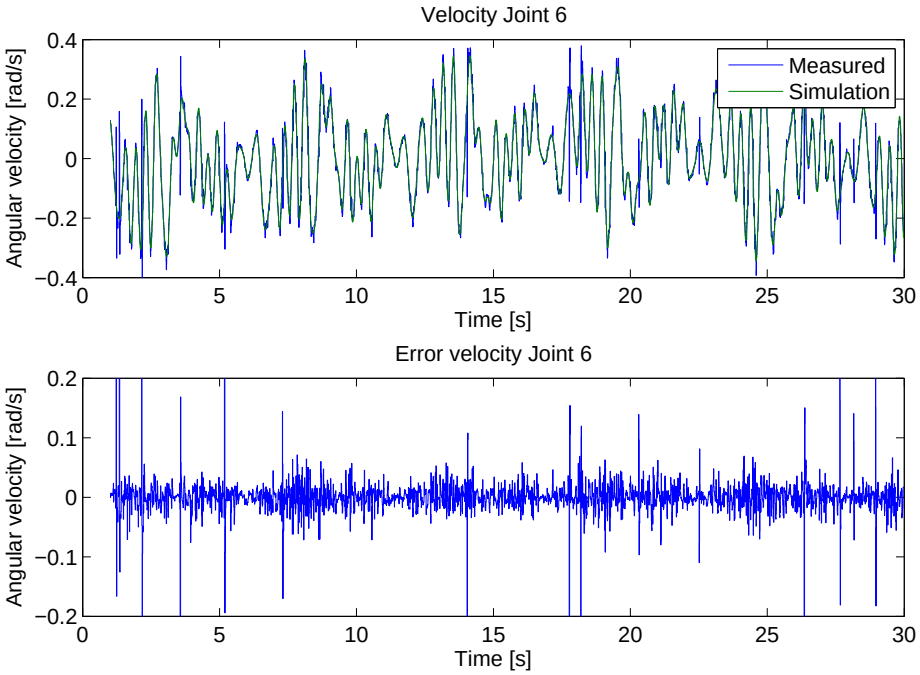


Figure B-6: Validation results for the Wrist3 Joint

Appendix C

Joint trajectory plots

In this Appendix the plots for the individual joints are depicted. It concerns a trajectory in x-direction, according to Figure C-1. The velocity is 10 mm/s. The method of control is Model Predictive Control (MPC) with the use of a Increment-Input-Output (IIO) model. The input is the velocity for the joints.

In the figures the position and the velocity of the individual joints are plotted. Furthermore the absolute error for these positions and velocities and the input is plotted.

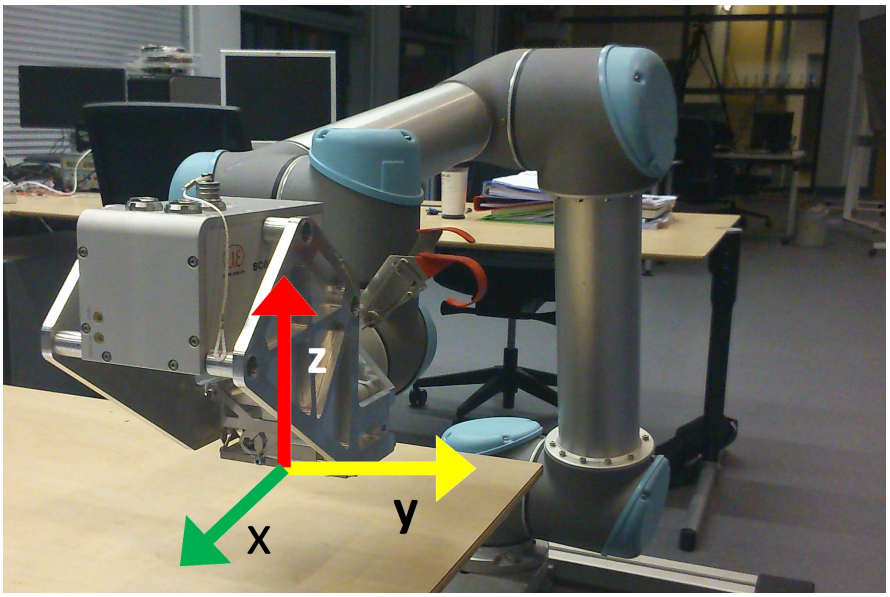


Figure C-1: Picture of the directions of the trajectory

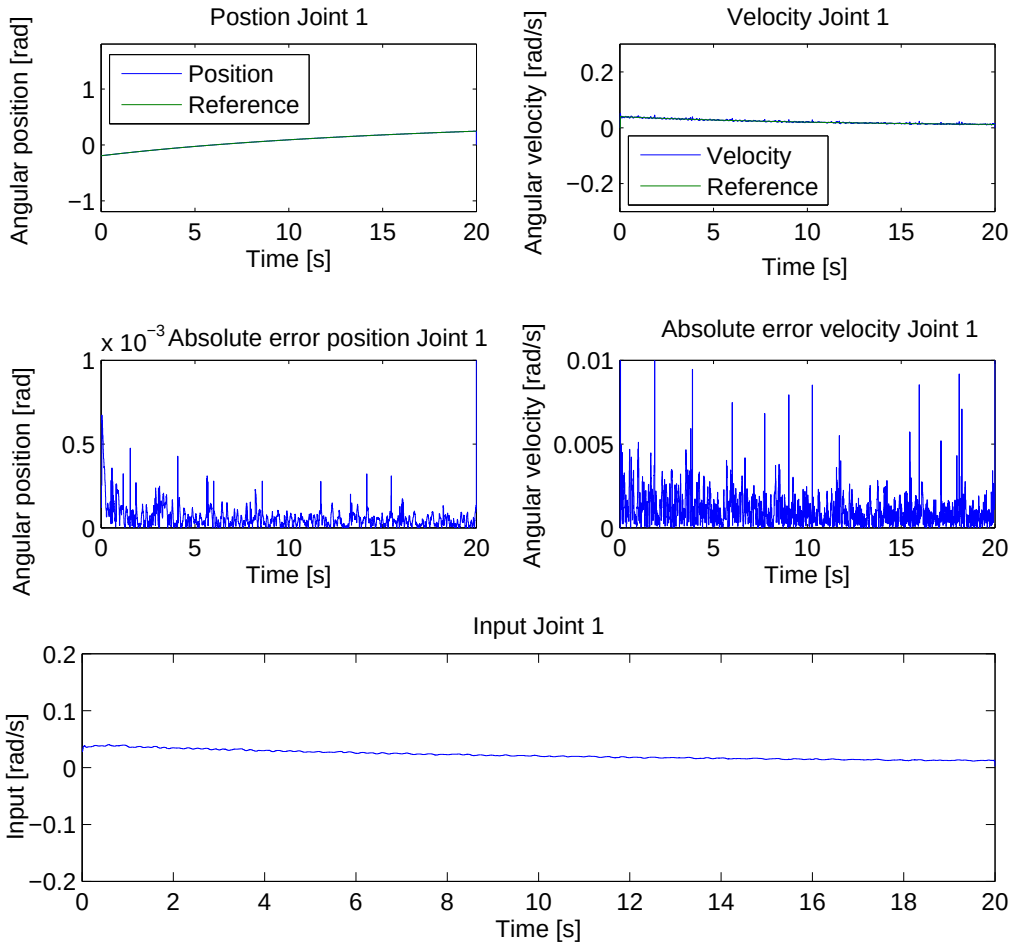


Figure C-2: Results for the Base Joint

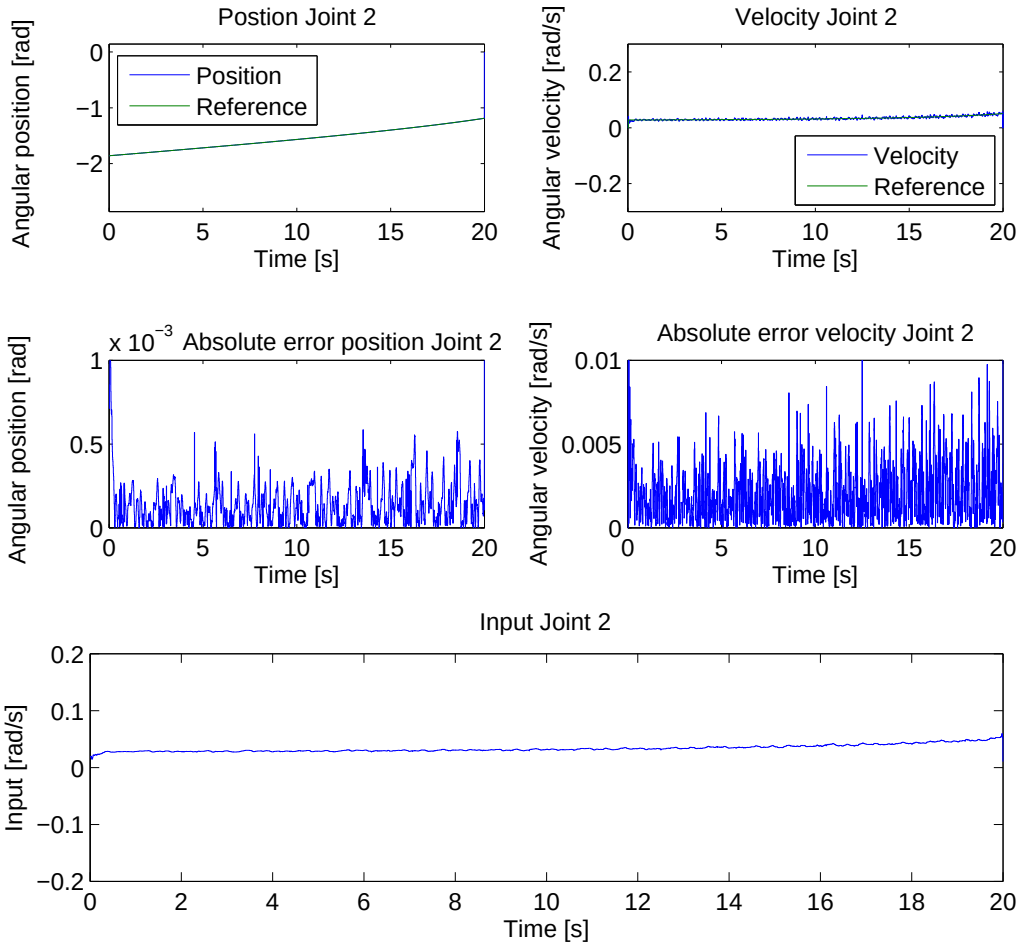


Figure C-3: Results for the Shoulder Joint

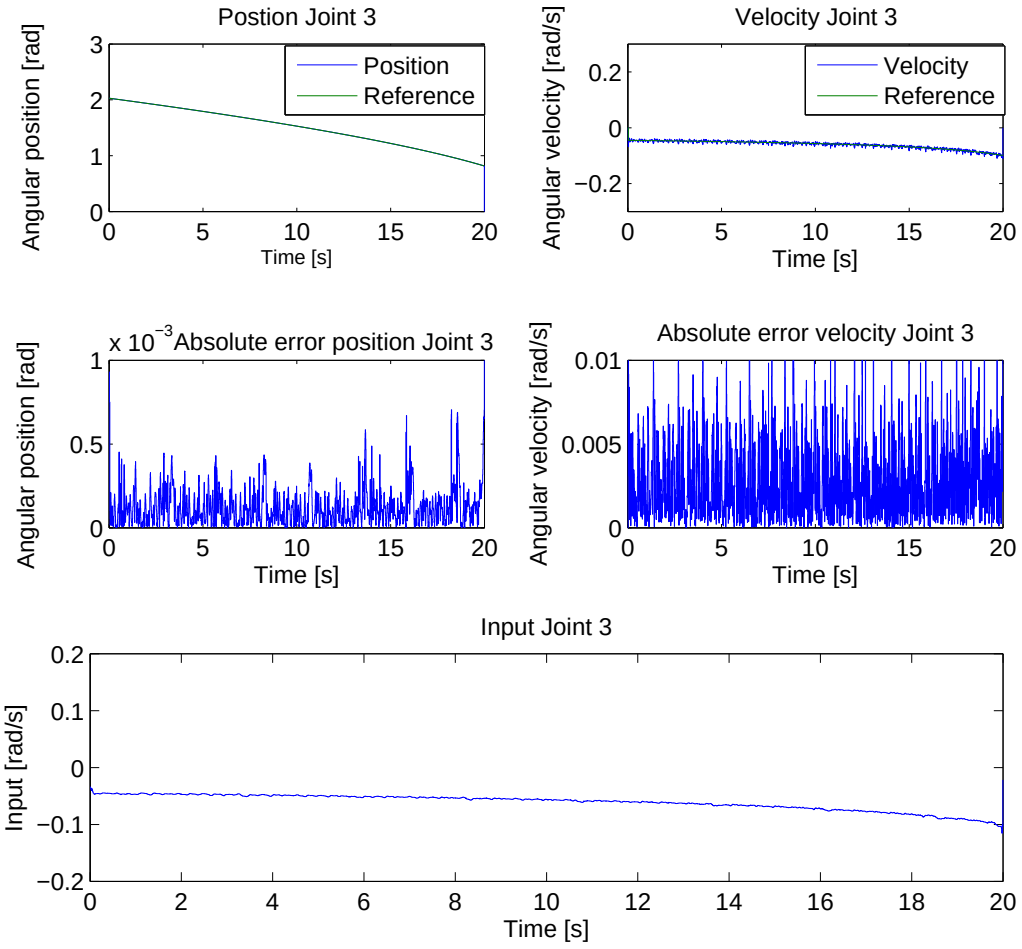


Figure C-4: Results for the Elbow Joint

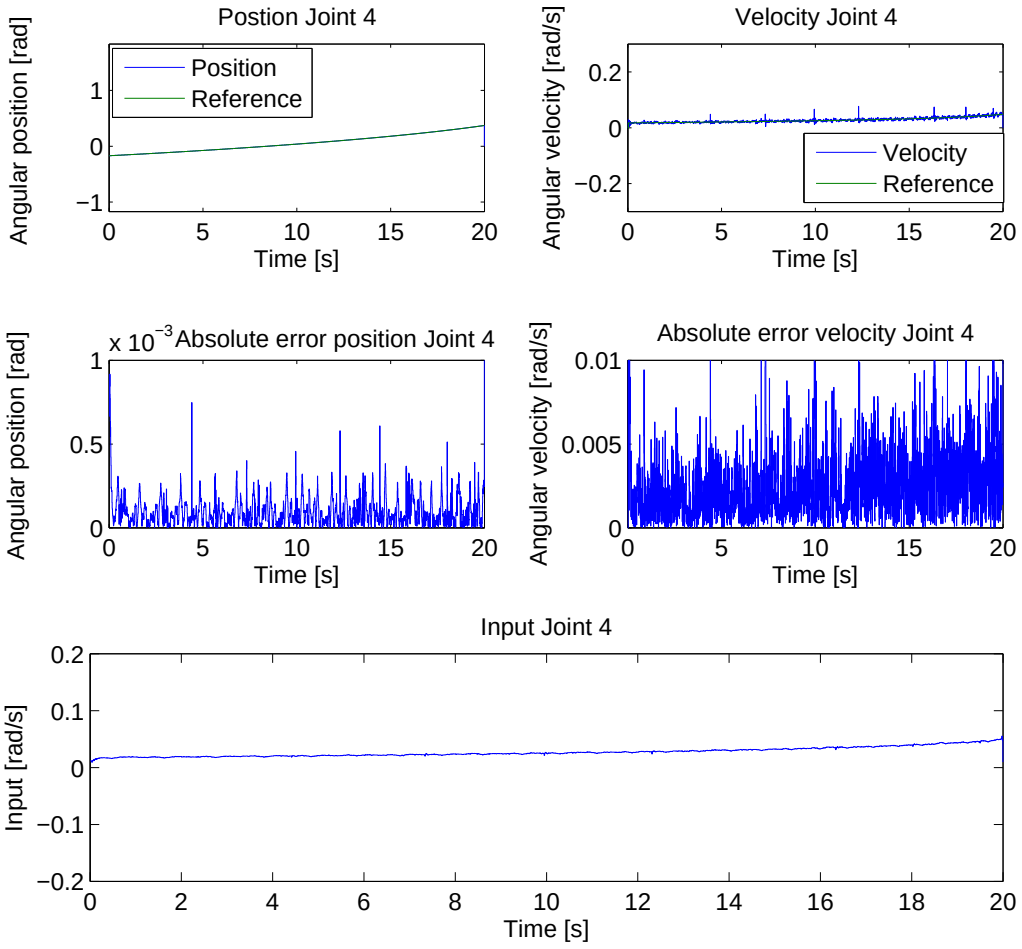


Figure C-5: Results for the Wrist1 Joint

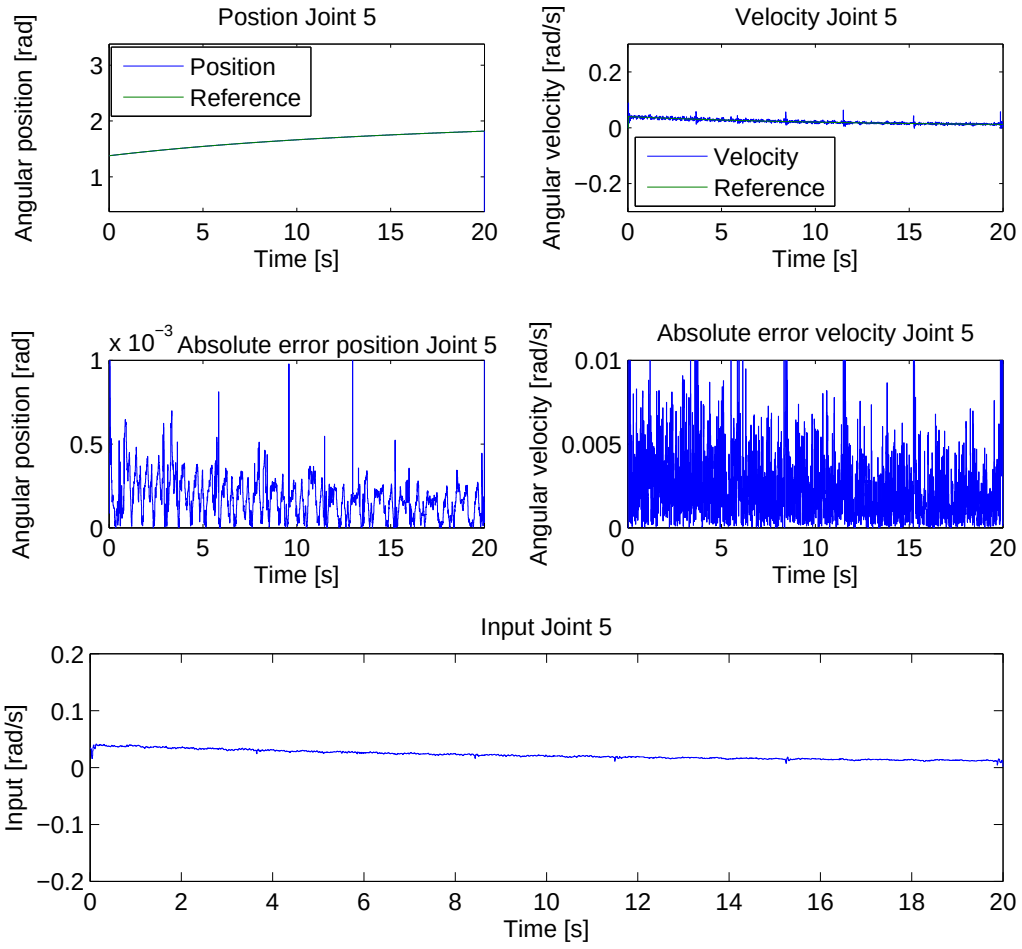


Figure C-6: Results for the Wrist2 Joint

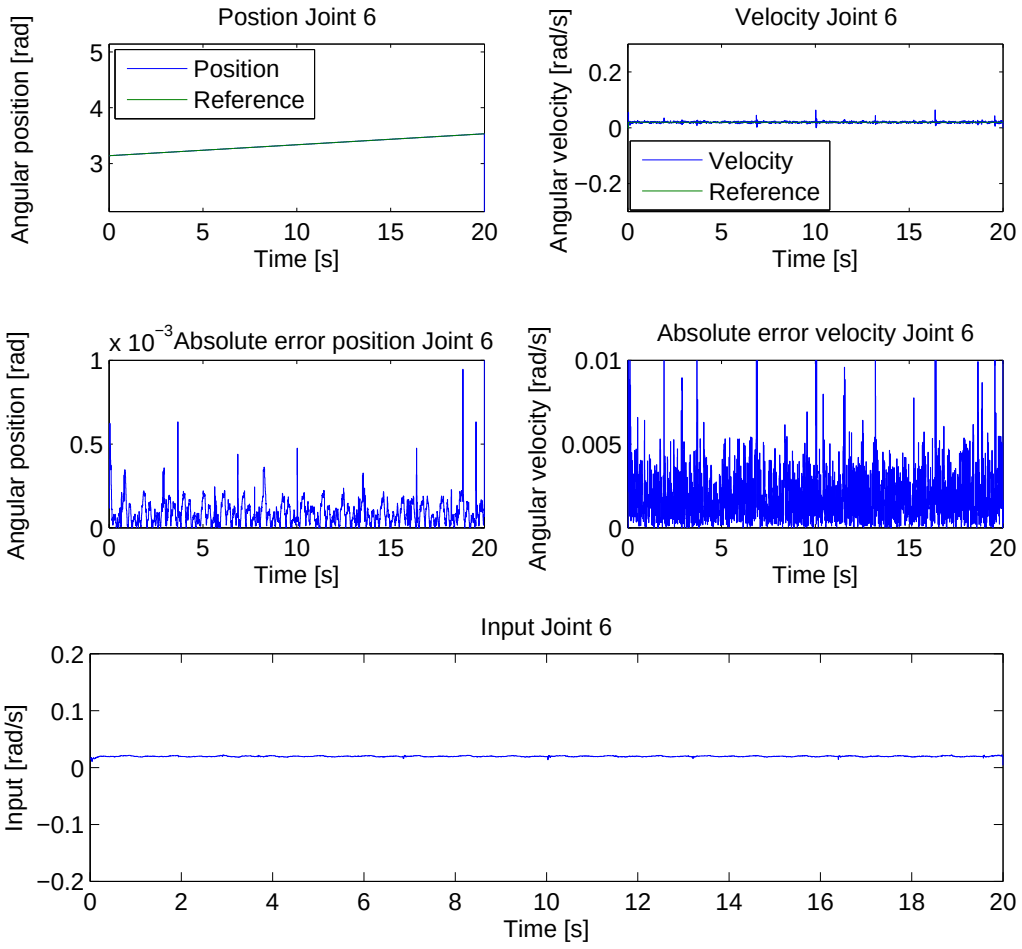


Figure C-7: Results for the Wrist3 Joint

Bibliography

[1] T. RowePrice, "A brief history of 3d printing," 2012.

[2] T. Economist, "The printed world," *The Economist Newspaper Limited*, vol. Feb 10, 2011.

[3] K. Yandell, "Organs on demand," *http://www.the-scientist.com/?articles.view/articleNo/37270/title/Organs-on-Demand/*, 2013.

[4] L. Nickels, "World's first patient-specific jaw implant," *Metal Powder Report*, vol. 67, no. 2, pp. 12 – 14, 2012.

[5] C. Kruit, "A novel additive manufacturing approach using a multiple degrees of freedom robotic arm," 2013.

[6] S. A. van Ipenburg, "3d printing technique for double curved surfaces using a robotic arm," 2014.

[7] J. P. Gazeau et al., "New printing robot for high-resolution pictures on three-dimensional wide surfaces," 2011.

[8] G. Boaz, "Continuous flow inkjet utilized for 3d curved surface printing," Feb. 26 2004. WO Patent App. PCT/IL2003/000,047.

[9] J. Lacaze, "Apparatuses for printing on generally cylindrical objects and related methods," Nov. 25 2010. WO Patent App. PCT/US2010/035,819.

[10] R. Uptergrove and M. Senta, "Apparatus and method for printing on articles having a non-planar surface," June 11 2013. US Patent 8,459,760.

[11] X. Chen et al., "Model optimization and image compensation in 3d printing," 2011.

[12] O. Pabst et al., "Inkjet printing and argon plasma sintering of an electrode pattern on polymer substrates using silver nanoparticle ink," *International Conference on Digital Printing Technologies*, vol. 26, pp. 146–149, 2010.

[13] D. L. Chandler, "Printing off the paper, mit research continues to push the boundaries of the burgeoning technology of 3-d printing," 2011.

[14] "Should we fence the arms of universal robots?." http://coro.etsmtl.ca/blog/?p=299.

[15] S. Stramigioli, "Modern robotics: Lecture slides set 1," 2010.

[16] J. Denavit et al., "A kinematic notation for lower pair mechanisms based on matrices," 1955.

[17] R. L. Williams II, "Notesbook supplement for me 4290/5290 mechanics and control of robotic manipulators," pp. 71–72, 2014.

[18] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Pearson Education International, 2005.

[19] A. Djuric, M. Filipovic, and L. Kevac, "Graphical representation of the significant 6r kuka robots spaces," in *Intelligent Systems and Informatics (SISY), 2013 IEEE 11th International Symposium on*, pp. 221–226, Sept 2013.

[20] M. Vaezi, H. Jazeh, F. Samavati, and S. Moosavian, "Singularity analysis of 6dof stäubli tx40 robot," in *Mechatronics and Automation (ICMA), 2011 International Conference on*, pp. 446–451, Aug 2011.

[21] S.-L. Wang and K. J. Waldron, "A study of the singular configurations of serial manipulators," *Journal of Mechanical Design*, vol. 109, pp. 14–20, 2008.

[22] M. Hayes, M. Husty, and P. Zsombor-Murray, "Singular configurations of wrist-partitioned 6r serial robots: A geometric perspective for users," 2002.

[23] D. Paganelli, "Topological analysis of singularity loci for serial and parallel manipulators," 2008.

[24] D. M. S. Gottschalk, M.C. Lin, "Obb-tree: A hierarchical structure for rapid interference detection," 1996.

[25] E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *Robotics and Automation, IEEE Journal of*, vol. 4, pp. 193–203, Apr 1988.

[26] D. Eberly, "Dynamic collision detection using oriented bounding boxes," *Geometric Tools, LLC*, 1999.

[27] W. Bittle, "Sat (separating axis theorem)." http://www.dyn4j.org/2010/01/sat/, 2010.

[28] S. J. Qin, "An overview of subspace identification," *Computers & Chemical Engineering*, vol. 30, no. 10âĂŞ12, pp. 1502 – 1513, 2006. Papers form Chemical Process Control {VII} {CPC} {VII} Seventh international conference in the Series.

[29] J. van Wingerden and M. Verhaegen, "Subspace identification of multivariable lpv systems: a novel approach," in *Computer-Aided Control Systems, 2008. CACSD 2008. IEEE International Conference on*, pp. 840–845, Sept 2008.

[30] S. Goto, ed., *Robot Arms*. Intech, 2011.

[31] "Orocos kinematics and dynamics." http://www.orocos.org/kdl/user-manual.

[32] "Openrave ikfast." http://openrave.org/docs/0.8.2/openravepy/ikfast/.

[33] K. Hawkins, "Analytic inverse kinematics for the universal robots ur-5/ur-10 arms," 2013.

[34] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011.

[35] J. A. Rossiter, *Model-Based Predictive Control: A Practical Approach*. CRC Press, 2003.

[36] B. d. S. Ton van den Boom, *Lecture Notes for the Course Optimization in Systems and Control*. 2011.

[37] T. J. J. van den Boom, *Lecture Notes for the Course Model Predictive Control*. 2013.

Glossary

List of Acronyms

2D	two dimensional
3D	three dimensional
AM	Additive Manufacturing
ARX	Autoregressive model with exogenous inputs model
ARMAX	Autoregressive-moving-average model with exogenous inputs model
DCSC	Delft Center for Systems and Control
DH	Denavit-Hartenberg
DOF	Degrees Of Freedom
ERMS	Error Root-Mean-Square
FDM	Fused Deposition Modelling
GJK	Gilbert - Johnson - Keerthi
IIO	Increment-Input-Output
IO	Input-Output
IPB	Ink Powder Binding
MIMO	Multiple-Input Multiple-Output
MPC	Model Predictive Control
OBB	Oriented Bounding Box
ROS	Robot Operating System
RP	Rapid Prototyping

SAT	Separating Axis Theorem
SID	Subspace Identification
SISO	Single-Input Single-Output
SLA	Stereo Lithography
TCP	Tool Center Point
VAF	Variance Accounted For