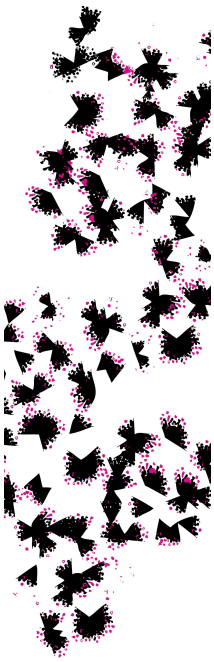

Controller Design for a Rail Mounted Inspection Robot Manipulator

L.M. (Laurie) Overbeek

MSc Report

Committee:
Prof.dr.ir. S. Stramigioli
D.J. Borgerink, MSc
Dr. R. Carloni
Dr.ir. R.G.K.M. Aarts
Dr.ir. J.F. Broenink

November 2015

034RAM2015
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

1 Introduction

Inspecting ballast water tanks on ships is a dangerous job for the inspectors. Currently research is done to see if this can be done autonomously in the future. A team of 7 partners worked on the project called "RoboShip" in which a rail-guided robot was designed that is capable of inspecting water ballast tanks. A manipulator has been designed to inspect the interior walls of the ballast water tank.

Currently the manipulator can only be steered with a human in the loop. This master thesis focuses on the design of an autonomous controller for the manipulator. The design and implementation is the main part of this report and is written in the form of an academical paper.
In addition, three appendices are included. In Appendix A the model is presented which has been used to test different control strategies. Forward kinematics is used to determine the error between the end-effector and the setpoint. To verify the forward kinematics, experiments have been done using the OptiTrack system, the results are shown in Appendix B. Obstacle avoidance is not yet implemented for the manipulator. However for a cluttered environment, like a ballast water tank, it is very important that this will be included in the future. Some recommendations for obstacle avoidance have been written down in Appendix C.

Contents

1 Introduction **I**

2 Paper **1**
1 Introduction . . . 1
1.1 Problem statement . . . 2
1.2 Goal . . . 2
1.3 Outline . . . 2
2 Controller for the RoboShip manipulator . . . 2
2.1 RoboShip manipulator setup . . . 2
2.2 Controller choice and requirements . . . 2
2.3 Overview proposed controller . . . 3
2.4 Impedance controller . . . 3
2.5 Error calculation . . . 4
2.6 Gravity compensation . . . 4
2.7 Complete control law . . . 4
3 Implementation . . . 5
3.1 Experimental setup . . . 5
4 Choice of loop controller . . . 5
4.1 Results and evaluation loop controller . . . 5
5 Experiments . . . 6
6 Results . . . 6
6.1 Experimental results . . . 6
7 Discussion . . . 8
8 Conclusion and recommendations . . . 9
8.1 Conclusion . . . 9
8.2 Recommendations . . . 9

Appendix **11**
A Modelling the manipulator of the RoboShip robot . . . 11
A.1 Dynamic and kinematic model . . . 11
A.2 Joint submodel . . . 12
A.3 Complete model of the manipulator . . . 12
B Kinematics verification using OptiTrack . . . 15
B.1 Method . . . 15
B.2 Optimized actuator offsets . . . 15
B.3 Area calculation . . . 16
B.4 Discussion . . . 16
C Obstacle avoidance . . . 19
C.1 Recommended approach . . . 19
C.2 Modeling . . . 19
C.3 Minimal distance calculation . . . 19
C.4 Foreseen problems and possibilities . . . 20

Controller Design for a Rail Mounted Inspection Robot Manipulator

L.M. Overbeek, D.J. Borgerink, J.F. Broenink and S. Stramigioli

Abstract

This paper focuses on the controller design of the RoboShip manipulator. The manipulator has been designed to inspect ballast water tanks autonomously. The goal of this paper is to design and implement the controller on the manipulator that is able to steer the end-effector to a predefined setpoint.
The manipulator needs to remain stable when the end-effector contacts the wall of the ballast water tank to take a measurement. The designed controller is implemented on the RoboShip manipulator and tested in both free space and in contact with the environment. Experiments indicate that the manipulator is capable of operating both in free space and in contact with the environment.
Keywords: Operational space, redundant manipulator, ballast water tank, impedance control, PID-controller, gravity compensation

1 Introduction

Ballast water tanks (BWTs) are used to stabilize ships. This is done by controlling the amount of water in the tanks. Salt water is corrosive and harms the steel ballast tanks. Therefore, a crew of inspectors is legally obliged to inspect the inside of a tank regularly for damage. Up to this moment this is done manually through visual inspection and handheld inspection devices, like thickness sensors. This inspection has to take place in a dry dock [1] and the ship needs to be taken out of operation. The inspection of BWTs is a dangerous and costly task so it would be advantageous if this could be done by robots instead of humans in the future.
Potentially suitable robotic concepts for automated maintenance of BWTs include flying robots [2], magnetic crawlers [3] [4], legged robots [5], robofish [6] and cable-guided robots [7]. All of these concepts have drawbacks in view of the structure of the tanks, limited payload or vulnerability to contamination in the tank. In previous studies, Christensen et al. [8] showed that a rail-guided robotic system has the greatest potential, therefore, this paper focuses on a rail-guided robot. This robot requires a manipulator to position a sensor on the interior walls of the BWT.
Borgerink [9] has shown that for the manipulator a single serial manipulator clamped on a compliant rail is not desirable. A different design that contacts the wall with an intermediate end-effector was suggested. This will increase the stability and relax the demands posed on both the rail and the robot. The designed robotic arm consists of a large-stroke, designed by Huttenhuis [10], and a small-stroke to reach the setpoint with more precision, see Figure 1. The large-stroke (the intermediate end-effector) needs to be maneuvered to the setpoint. This intermediate end-effector will latch on to the wall of the BWT using an electromagnet. The small-stroke (with the sensor) can then be positioned with more precision at the point of interest. The RoboShip manipulator is a 6 DOF manipulator for a 5 DOF task space. The task space is 5 DOF since the end-effector has to be placed perpendicular to the tank-wall, the rotation around this orthogonal axis is irrelevant. The small-stroke has a workspace with a radius of 10 cm. To be able to place the end-effector on the tank-wall with enough precision a controller is designed and implemented on the 6 DOF manipulator.

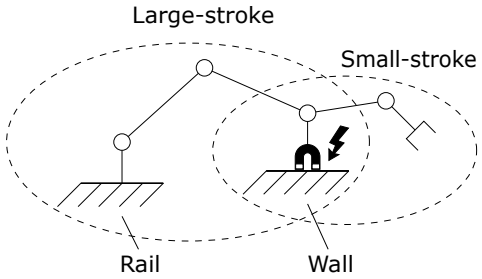

Figure 1: Concept of the two stage manipulator

An initial controller that was designed for the manipulator is a joint-space controller. To steer the manipulator to a certain setpoint a couple of waypoints were generated. These waypoints consisted of the joint angles and were generated by moving the arm manually along a trajectory. This is unde-

sirable because an operator would have to generate all these waypoints for every possible setpoint in the BWT. Another problem with this method is that, due to gravity the arm was unable to reach the desired angles and thus the exact setpoint. The internal PID controller of the present actuators is unable to cope with any I-action to compensate for friction and gravity. The actuators will give an overload error if the I-action is not set to zero. Therefore, this control strategy will not work on this setup.

Another implemented method was to steer the manipulator to a given setpoint using an operator and a 3-D mouse. The operator would move the mouse in a certain direction, which is translated to an end-effector twist. This twist was then transformed to angular velocities by using the pseudo-inverse Jacobian. In this case the accuracy that the manipulator was able to reach depended on the operator. In an uncluttered environment this could work. However, when an operator has to steer the serial manipulator in a confined space with obstacles it is very difficult to imagine what consequences an end-effector twist has on all the different links. Both implementations require a human in the control loop. This is unwanted, therefore a new control strategy needed to be developed where the human is taken out of the loop.

1.1 Problem statement

The above mentioned controller implementations require a human in the control loop. This is unwanted and therefore a new control strategy is developed. The goal is to design an autonomous controller, such that the human is taken out of the loop and that is able to compensate for gravity and friction.

1.2 Goal

The primary purpose of this paper is to present a controller for the manipulator of the Roboship robot. This paper will focus on the control of the large-stroke. The second purpose is to implement the controller and give a proof of concept.

1.3 Outline

This paper is organized as follows. In Section 2 the controller designs are presented. Then Section 3 describes the setup and in Section 4 a PD and PID controller are compared. Section 5 describes the experiments on the chosen controller. The experimental results of the controller are presented in Section 6. A brief discussion about the found results is given in Section 7 and in Section 8 the conclusions and recommendations will be given.

2 Controller for the RoboShip manipulator

In this section a brief analysis of the controller that is implemented is given and the theory behind the controller is explained.

2.1 RoboShip manipulator setup

Every joint in the RoboShip manipulator is an actuated joint. Joint 1 through 5 are equipped with a Dynamixel MX-106 servo motor. For joint 6 (near the end-effector) a different motor was chosen. This joint is equiped with the Dynamixel MX-28 because it weighs less and it needs to deliver less torque than the other joints. Joint 2 is equipped with two actuators, one acting as master and the other as slave, such that it can deliver more torque to lift the manipulator. The Dynamixel MX-106 can be position controlled or torque controlled. The Dynamixel MX-28 only has the option to be position controlled. When a torque controller is selected this actuator can be controlled by using:

$$\Delta\ q_6 = \alpha\ \tau \tag{2.1}$$

2.2 Controller choice and requirements

The manipulator has to reach a setpoint in space that is defined by a position and orientation. The end-effector has to be orthogonal to the wall of the BWT. How it is rotated around the orthogonal axis when at its setpoint is irrelevant to be able to take a measurement. A virtual joint is used at the end-effector that can rotate around the orthogonal axis. This is done to reshape the control problem to a 7 DOF manipulator with a 6 DOF task space [14]. There are two options to control the RoboShip manipulator, it can either be controlled by setting desired angles (joint space control) or by torque control. To calculate the desired angles an inverse kinematics algorithm can be used. This manipulator is a redundant manipulator, since the task space is only 5 DOF and the manipulator has six joints. This means that it is possible that there exist multiple solutions to the inverse kinematics algorithm. To inspect the BWT the configuration of the manipulator is not of any importance. The position and orientation of the end-effector are important. If the desired angles are calculated using inverse kinematics and one or more joints are not able to reach their setpoint the error of the end-effector

cannot be compensated by a different joint. For this reason an operational controller is preferred.

A few options are available when choosing for an operational-space torque controller: computed torque control and impedance control. Impedance control is often used when the manipulator operates both in free space and in contact with the environment, described or predicted in the absence of a complete characterization of both systems. For computed torque control an exact model of the manipulator is required to be able to calculate the needed torques. Because it is impossible to make an exact model of both the environment and robot it is chosen to implement an impedance controller.

Friction is a problem in the RoboShip manipulator, therefore a compensation term can be introduced to account for this. Feed-forward friction compensation is a solution to avoid an integral action in the controller. To do this properly a friction profile needs to be made of every motor (in this case seven). The motors in this setup are not able to handle the load reliably. This causes wear in the motors which alters the friction profile. Hence, this is not a solution for the RoboShip manipulator. An integral action will be used instead.

To be able to operate in both free space and in contact with the environment the control law is switched before it touches the environment to decrease the chance that the manipulator becomes unstable.

2.3 Overview proposed controller

The proposed controller consists of three stages. The first stage is an impedance controller with stiff springs for more accuracy. When the end-effector reaches the steady state there will be an error left due to friction. To reduce the steady state error, the second stage starts and adds an integral action. The integral action is added in a later stage to prevent windup from the large initial error.
The third stage is when the controller is switched to a more compliant impedance controller and the integral action is frozen. This is done so that the manipulator remains stable when the electromagnet pulls the manipulator towards the wall of the BWT and to prevent unwanted control actions. The control law is switched as soon as the end-effector is within the margin of error, which is set to a translational error of 2 cm, measured as euclidean distance to the setpoint, and a rotational error of 10 degrees. These values for the margin of error are chosen since the electromagnet is able to pull the end-effector towards the wall from this distance. The output of the integral action right before the switch is given as feed-forward signal to compensate for friction in the joints and errors in the estimation of the gravity compensation and is added as a constant torque to the controller. All three stages use gravity compensation to compensate for the gravitational forces acting upon the manipulator.

2.4 Impedance controller

The dynamic equation of an open chain serial manipulator is given by:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau_{tot} \tag{2.2}$$

Where $M(q)$ is the inertia matrix, $C(q,\dot{q})$ represents the Coriolis and centrifugal terms, $G(q)$ is the gravitational term and τ_{tot} is the equivalent of all joint forces on the manipulator. If the wrench is known that the spring applies to the end-effector, the necessary torques on the joints can be calculated with:

$$\tau^T = J^T(q)\ (W_S^{0,e})^T \tag{2.3}$$

Where J is the geometrical Jacobian and $W_S^{0,e}$ is the wrench that the spring applies to the end-effector. To make the controller more stable in the presence of a stiff spring a damper is added. This damper can be added in joint space, the control law then becomes:

$$\tau_{tot}^T = J^T\ (W_S^{0,e})^T - G_d\ \dot{q} \tag{2.4}$$

Where G_d is the damping gain and $\dot{q}$ is the angular velocity, this is shown in Figure 2.

The wrench that the spring applies on the end-effector can be calculated with the following equations [11]:

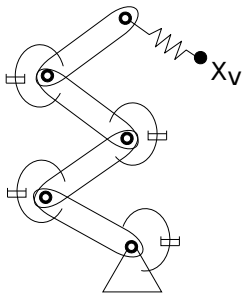

Figure 2: Multidimensional impedance control with spring and damping

$$
\begin{aligned}
W_S^{0,e} &= (m^e f^e) \\
\tilde{m}^e &= -2as(G_0 R_e^d) - as(G_t R_d^e \tilde{p}_e^d \tilde{p}_e^d R_e^d) \\
&\quad - 2as(G_c \tilde{p}_e^d R_e^d) \\
\tilde{f}^e &= -R_d^e as(G_t \tilde{p}_e^d) R_e^d - as(G_t R_d^e \tilde{p}_e^d R_e^d) \\
&\quad - 2as(G_c R_e^d)
\end{aligned} \tag{2.5}
$$

Where G_t, G_o and G_c represent the translational, orientational and coupled spring co-stiffnesses respectively. R_d^e and p_d^e are the rotational and translational parts of H_d^e. Where H_d^e is the homogeneous matrix between the end-effector position and the desired position.

When the manipulator operates in free space and outside of the margin of error a stiff configuration is used to get a higher accuracy. Inside the acceptable range the spring constant is reduced significantly to make the manipulator more compliant with the environment.

2.5 Error calculation

To prevent steady state errors due to friction an integration term is introduced. For the integral action the following error vector is used:
$e = \begin{pmatrix} e_p \\ e_o \end{pmatrix}$.
Where e_p denotes the position error and e_o the orientation error.
The position error is given by $e_p = p_d - p_e$ where p_d and p_e respectively denote the desired and actual end-effector positions. To determine the orientation error the axis-angle method is used since it describes the error in a geometrical way. In [16] it is shown that the orientation error can be expressed as:

$$e_o = u_r \ sin(\theta) \tag{2.6}$$

where u_r is the normalized axis and the angle is θ, this is shown in Figure 3.

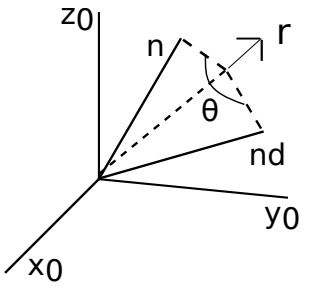

Figure 3: Orientation error

The normalized axis is obtained from the rotation matrix R_d^e. From equation (2.6) it can be seen that the solution is unique for the interval $-\pi/2 < \theta < \pi/2$.

2.6 Gravity compensation

Gravity compensation is a way to cancel the effects of gravitational force on the links. The link positions q are used to compute the torques necessary for gravity compensation. In Appendix A a model of the joint and link is shown. The kinematic structure of these models are used to calculate the forces due to gravity. An example of the model structure used to compensate for gravitational forces is shown in Figure 4. In the link the source of effort represents the gravitational force that is applied in the center of mass of the link. The joint is modeled as a zero junction with a torque sensor to measure the torque in that joint due to gravity. Using this structure the expected effort in each 1-junction and 0-junction can be calculated. The torque of the 0-junction in the DOF of the joint is the effort due to gravity that needs to be compensated.

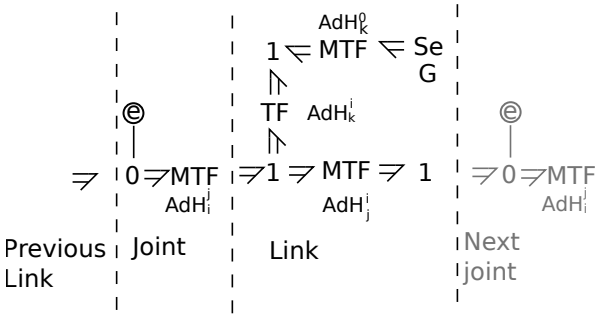

Figure 4: Model structure used to calculate the gravity compensation

2.7 Complete control law

The complete control law as described above, in free space, can be written as:

$$\tau^T = J^T \ (W_S^{0,e})^T - G_d \ \dot{q} + J^T \ G_i \int e \ dt + G(q) \tag{2.7}$$

Where e represents the error vector, G_d and G_i represent gains for the derivative and integral terms and $G(q)$ represents the gravity compensation. The integral action is only applied if the PD-controller has reached the steady-state error.
Near the wall the control law is switched to:

$$\tau^T = J^T \ (W_S^{0,e})^T - G_d \ \dot{q} + G(q) + I_{constant} \tag{2.8}$$

$W_S^{0,e}$ is calculated using weaker spring constants to make the controller more compliant. The term $I_{constant}$ represents the constant output of the integral action.

3 Implementation

In this section the setup is presented on which the controller is implemented to validate that the designed controller works correctly. To verify that the controller can reach a given setpoint several experiments are executed.

3.1 Experimental setup

All actuators of the RoboShip manipulator are connected to a computer using a serial communication protocol. The baud rate of the actuators is set to 200 kb/s. The controller is implemented in MATLAB R2014b and the update frequency towards the actuators is 16 Hz. This was the highest possible update frequency due to limitation caused by the time it takes to communicate with all actuators. The manipulator is mounted on a rail in such a way that it can reach setpoints above and below the rails. The setup can be seen in Figure 5. The error is determined by measuring the actuator angles and using forward kinematics to determine both the position and orientation of the end-effector.

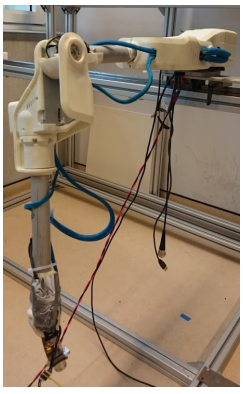

Figure 5: Experimental setup

4 Choice of loop controller

From previous implementations it is expected that the friction needs to be compensated. To test if an integral action is necessary to compensate for friction, a PD controller is compared with a PID controller. Preferably an integral action is avoided to decrease the risk of instability and windup. To prevent unnecessary and unwanted control actions, the position and orientation of the manipulator will be kept constant if it is within the margin of error.

The translational error of the end-effector is given by the euclidean distance between the end-effector position and the position of the setpoint. The angles of the actuators are measured; by using forward kinematics the position error of the end-effector is determined. For the rotational error it is only of importance that the end-effector is orthogonal to the wall. The frames of the setpoint and end-effector are chosen such that the y-axis needs to be aligned. When both axis are aligned the value of $R_e^d(2,2)$ equals 1. The orientation error is calculated using:

$$Orientation\ error = cos^{-1}(R_e^d(2,2)) \qquad (4.1)$$

4.1 Results and evaluation loop controller

In Figure 6 the error in position of the PD and PID controller can be seen. In Figure 7 the corresponding error in the orientation is given.

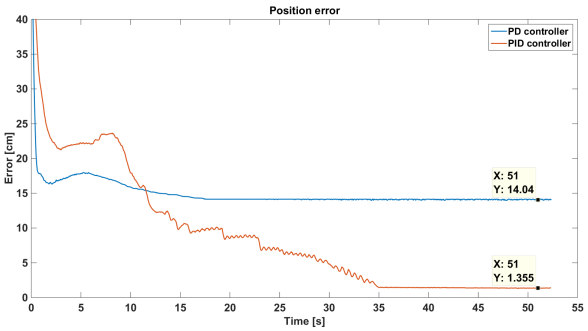

Figure 6: Normalized position error

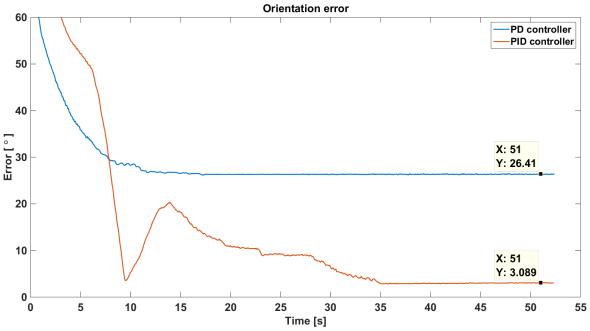

Figure 7: Orientation error

In Figure 6 it can be seen that for the PD controller the position error is larger than the allowed 2 cm.

The internal friction of the manipulator prevents the end-effector from reaching the desired setpoint. The PID controller has much better results.
When looking at the orientation error, see Figure 7, it can be seen that the error of the PD controller does not reach the desired margin. The PID controller does succeed in reducing the error to within the specified margin.
From this it is concluded that an integral action is needed to correct for friction. Only the PID controller will be considered from this point onwards.

5 Experiments

Several experiments have been performed to test the performance of the PID controller.

The manipulator has been steered towards multiple setpoints, to validate that the controller performs well. Its trajectory is analyzed in Section 6.

To see if the gravity compensation works as expected the results of the PID controller with and without the gravity compensation are compared. The euclidean error, orientation error and the error in the x-, y- and z-direction will be compared.

The ultimate goal is for the manipulator to be able to inspect a setpoint on the wall of the BWT without becoming unstable. The proposed controller is verified to remain stable during interaction with the wall of a BWT by placing a metal plate near the end-effector as soon as the end-effector is within the margin of error. The magnet on the end-effector will then pull the manipulator towards the wall and lock it in place. The metal plate is introduced into the space when the manipulator is already within the margin of error because no obstacle avoidance has been implemented yet.

6 Results

This section presents the experimental results that are used to evaluate the proposed controller. The experiments discussed in this section are all performed on the RoboShip manipulator with the PID controller discussed in the previous sections. The parameters used in the controller when in free space are given in Table 1. The parameters used during interaction with the environment are given in Table 2. Where diag([ ]) represents a diagonal matrix, k_t, k_0 andk_c represent the translational, orientational and coupled spring constants respectively.

Parameters	**Free space stage**
k_t	diag([5,5,5])
k_o	diag([0.6,0.6,0.6])
k_c	0
I	diag([0.25,0.25,0.25,0.5,0.5,0.5])
D	diag([0.5,0.5,0.5,0.5,0.5,0.5])

Table 1: Parameters used during experiments in free space

Parameters	**Interaction stage**
k_t	diag([0.5,0.5,0.5])
k_o	diag([0.06,0.06,0.06])
k_c	0
I	output is kept constant
D	0

Table 2: Parameters used during interaction with the environment

6.1 Experimental results

PID controller

In Figure 8 the position error of the end-effector in the x-, y- and z-direction is shown. This figure shows that the movement of the manipulator is not smooth. Looking at the error in the z-direction it can be seen that the trajectory experiences multiple bumps. These bumps are caused by the internal friction of the manipulator. When the torque gets below a certain value, the friction of the motors will temporarily prevent the motors from continuing to move. Either the error has to increase sufficiently first or the I-action should become large enough to continue decreasing the error. Figure 9 shows the position and orientation error of the PID controller. In this figure it is shown that the both the position and orientation error are within the margin of error.

Integrating both the orientation and position error can have drawbacks. If the manipulator needs to change its orientation to decrease the rotational error it could be that one or more joints change their angles in such a way that it increases the position error, causing the two errors to interfere with each other. It is possible for the manipulator to stay temporarily in a local minimum when integrating both the position and orientation error. In Figure 10 it is visible that the error in the z-direction remains large and stable for an extended period of time. In this case for the manipulator

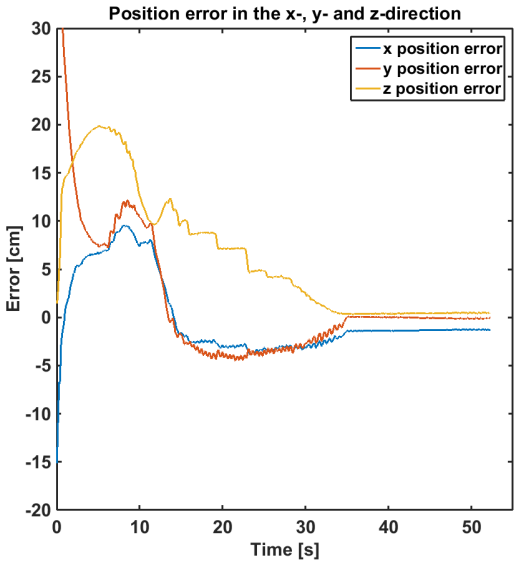

Figure 8: Position error of the end-effector in the x-,y- and z-direction

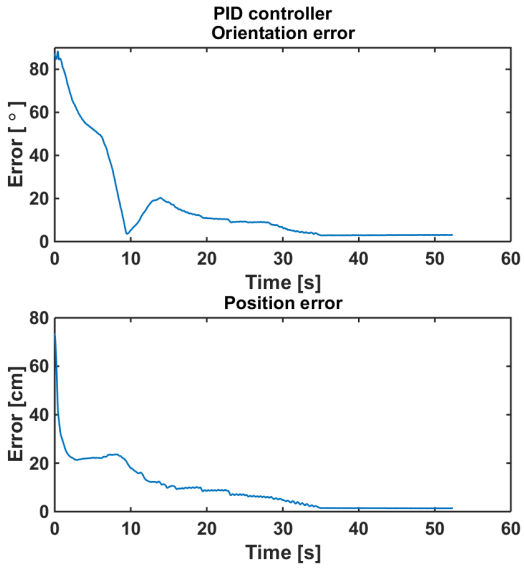

Figure 9: Position and orientation error

to decrease the position error, mainly the second joint should decrease its current angle, which can be achieved with a negative torque. However, when the controller tries to decrease the orientation error the torque needed in the second joint is positive as shown in Figure 11. In a worst case scenario this can also result in instability of the controller when the position and orientation errors increase and decrease alternating but never become zero.

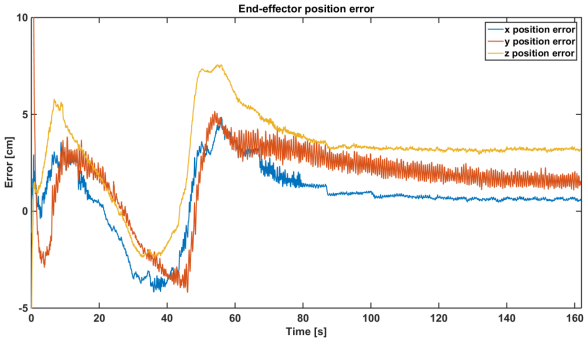

Figure 10: Manipulator stays in local minimum

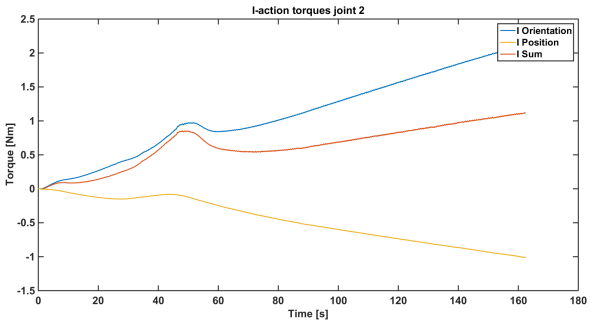

Figure 11: I-action for the position and orientation

Gravity compensation

In Figure 12 the error plot of both the position and orientation error is shown when using gravitational compensation compared to when not using this approach. The positon error in the x-, y- and z-direction of both the controller with and without gravity compensation of the end-effector can be seen in Figure 13. It can be seen that the position error plot of the PID controller with gravity compensation is much smoother and stabilizes much faster. This can be explained by larger error since links will "fall" due to gravity, the I action now has to take over the gravity compensation. This results in a larger integrated error, hence overshoot is more likely to occur when the control parameters are kept the same.

Three stage controller

The controller is switched to a more compliant controller when the manipulator is near the wall and the position error and orientation error are within the margin of error. The P controller gains are decreased by a factor 10 and the integral output is kept constant. Figure 14 shows that once the manipulator is inside the margin of error it remains at the same position. Once the metal plate is brought

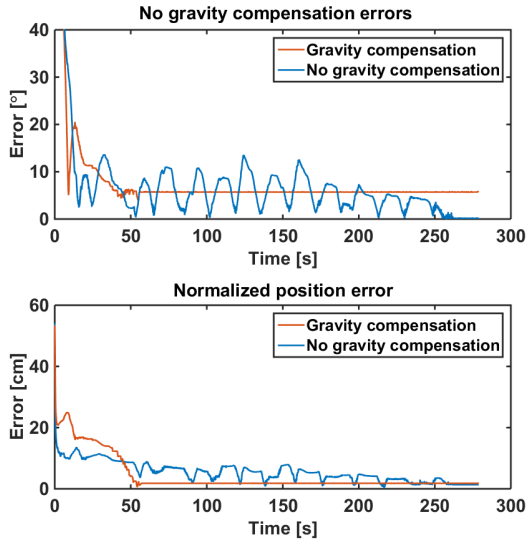

Figure 12: Error of the end-effector

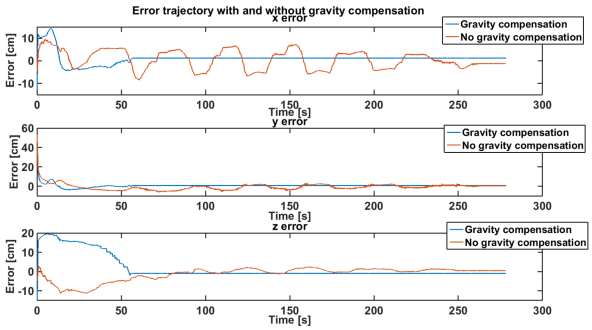

Figure 13: Position error of the end-effector in the x-, y- and z-direction

into close proximity of the end-effector the magnet starts pulling the end-effector towards the metal plate. As can be seen in this figure, even though the position error increases, due to the pull of the electromagnet, the manipulator remains stable. In this case the "position error" increases from 1 cm to 3 cm, as shown in Figure 15, which can still be reached with the small-stroke.

7 Discussion

The PID controller was able to steer the manipulator to the desired setpoints. Using the integral action also helped to compensate for friction. As showed in the results the PD-controller was not able to reach the desired setpoints due to the internal friction in the manipulator. Integrating both the orientation and position error can be conflicting

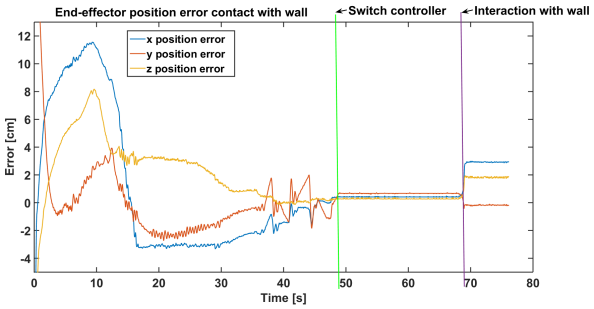

Figure 14: Interaction of the manipulator with the wall.

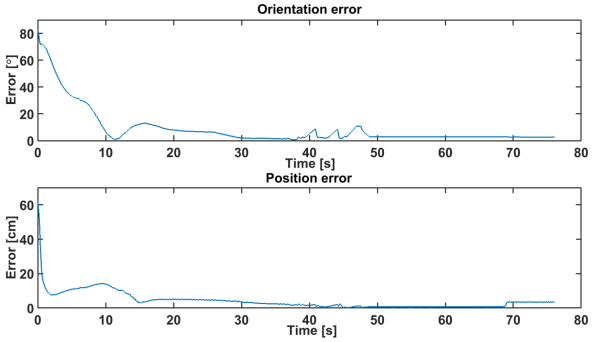

Figure 15: Position and orientation error

and can prevent the manipulator from reaching its setpoint. In a worst case scenario it can make the controller unstable when the orientation error and position error alternatingly decrease. Due to this the magnitude of the needed torque will increase and the controller will become unstable.

The three stage controller works as expected. The manipulator is able to reach the wall without becoming unstable, even when the error increased due to the pull of the electromagnet the controller functioned properly.

As could be seen in the results, adding gravity compensation to the control law greatly speeds up the trajectory of the manipulator to its desired setpoint while also at the same time enhancing the accuracy and stability.

Forward kinematics was used to determine both the position and orientation error. This is only correct if the forward kinematics are true. To verify this experiments where done which can be found in Appendix B.

During the experiments the Dynamixels suffered often from overload errors, the actuators have an in-

ternal protection for overload that sets the delivered torque to zero. When this happens the manipulator falls, increasing the error which is undesirable. The overload errors are caused because the Dynamixels cannot deliver enough torque to the manipulator. The Dynamixels in joint 2 often suffered from overheating problems (occurring when the actuator becomes $> 80\ ^\circ C$) which shuts down the motor to prevent damage. When such an error occurs the actuator needs to cool down. If this were to happen inside the tank during an inspection it would be a costly procedure. Besides the previous problems the setup also suffered from communication errors that were caused by a broken driver chip within the Dynamixel. This error has been fixed by replacing the faulty chip.

8 Conclusion and recommendations

8.1 Conclusion

The designed controller for the manipulator steers the end-effector to the desired setpoint. A PID controller was successfully used to compensate for the internal friction in the manipulator. In some situations the manipulator became unstable since integrating both the position and orientation error can lead to conflicting situations.

Adding gravity compensation to the controller made the trajectory of the end-effector much more accurate and smooth and it reaches its desired setpoint much faster.

The designed controller was able to switch the control law flawlessly. After the switch the manipulator stayed within the margin of error and when the magnet pulled the manipulator towards the wall the manipulator remained stable.

Due to the problems with the actuators the manipulator is not reliable enough and due to this it is unable to reach every setpoint in the workspace. The actuators in joint 2 cannot supply enough torque which caused overload/overheating problems.

8.2 Recommendations

The actuators in joint 2 were unable to deliver enough torque. Therefore, it is recommended to redesign the manipulator in such a way that other actuators can be used that can deliver more torque and preferable have less friction. At this moment when an overload error occurs, the internal software sets the torque setpoint to zero. When this happens no torque is delivered to the manipulator and it will fall down. This is an undesired situation and could harm the manipulator. For this reason it is recommended to use actuators in joint 2 that can deliver enough torque and of which the internal software can be adapted such that the torque is not set to zero and the manipulator will not fall uncontrolled.

The redundancy of the manipulator is not actively used at the moment. This redundancy can be used to avoid joint limits. This way the controller will still be able to reach the setpoint using different torques if it encounters a joint limit.

Another improvement for the control of the manipulator is to implement obstacle avoidance. At this stage the manipulator is unaware of the objects in close proximity. This caused the robot to collide with the rails during some of the experiments. To solve this problem it is recommended to use an obstacle avoidance algorithm combined with a path planning algorithm. A recommended approach to solve this problem can be seen in Appendix C.

References

[1] Christensen, L., Kirchner, F., Fischer, N., Ahlers, R., Psarros, G., & Etzold, L. (2011). Tank inspection by cost effective rail based robots. In Proceedings of International Conference on Computer Applications in Shipbuilding (ICCAS).

[2] Keemink, A. Q. L., Fumagalli, M., Stramigioli, S., & Carloni, R. (2012, May). Mechanical design of a manipulation system for unmanned aerial vehicles. In Robotics and Automation (ICRA), 2012 IEEE International Conference on (pp. 3147-3152). IEEE.

[3] Focusproject: Ship Inspection Root, ETH Zürcih, `http://www.sir.ethz.ch`,2013

[4] Vögele, T., Eic, M., Grimminger, F., & Fondahl, K. (2010). A Hybrid Legged Wheel Climbing Robot for Marine Inspection. In Proceedings of International Conference on Climbing and Walking Robots and the Support Tehnologies for Mobile Machines.

[5] Mazumdar, A., & Asada, H. H. (2009, October). Mag-foot: A steel bridge inspection robot. In Intelligent Robots and Systems,

2009. IROS 2009. IEEE/RSJ International Conference on (pp. 1691-1696). IEEE

[6] Liang, J., Wang, T., Wang, S., Zou, D., & Sun, J. (2005). Experiment of robofish aided underwater archaeology. In Robotics and Biomimetics (ROBIO). 2005 IEEE International Conference on (pp. 499-504). IEEE.

[7] Fang, L.J. & Wang, H.G., "Research on the motion system of the inspection robot for 500kV power transmission lines," in Applied Robotics for the Power Industry (CARPI), 2010 1st International Conference on , vol., no., pp.1-4, 5-7 Oct. 2010

[8] Christensen,L., Fischer,N., Kroffke,s. Lemburg, J. & Ahlers,R. "Cost Effective Autonomous Robots for Ballast Water Tank Inspection".

[9] Borgerink, D. J., Stegenga, J., Brouwer, D. M., Wortche, H. J.,& Stramigioli, S. (2014, September). Rail-guided robotic end-effector position error due to rail compliance and ship motion. In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on (pp. 3463-3468). IEEE.

[10] J.A.J Huttenhuis, D.J. Borgerink, D.M. Brouwer, R.G.K.M. Aarts & S. Stramigioli. Kinematic Design Method for a Rail Mounted Inspection Robot Arm. Master thesis 2015

[11] Stramigioli, Stefano, "Modleing and IPC control of Interactive Mechanicla Systems: a coordinate-free approach" 2001

[12] Hogan, Neville. "Impedance control: An approach to manipulation." American Control Conference, 1984. IEEE, 1984.

[13] Hogan, Neville and Buerger, Stephen, " Robotics and Automation: Chapter 19, Impedance and interaction control" ,CRC Press LLC, 2005.

[14] Baron, Luc. "A joint-limits avoidance strategy for arc-welding robots." Int. Conf. on Integrated Design and Manufacturing in Mech. Eng. 2000.

[15] Lee, Ho-Yul, Byung-Ju Yi, and Yungjin Choi. "A realistic joint limit algorithm for kinematically redundant manipulators." Control, Automation and Systems, 2007. ICCAS'07. International Conference on. IEEE, 2007.

[16] Luh, J. Y., Walker, M. W., & Paul, R. P. (1980). Resolved-acceleration control of mechanical manipulators. Automatic Control, IEEE Transactions on, 25(3), 468-474.

A Modelling the manipulator of the RoboShip robot

This appendix presents the dynamic model of the manipulator of the RoboShip. The model is used in order to evaluate different controllers for the manipulator. The modeling of a complex robotic systems is time consuming and error prone. Dresscher [1] designed a generalized power continues submodel that can be re-used for modeling robotic manipulators using the bondgraph language. These submodels are used to model the RoboShip manipulator.

A.1 Dynamic and kinematic model

The manipulator is a 6 DOF arm (large-stroke) with an additional 2 DOF (small-stroke) at the end-effector, that can be controlled separately to inspect the location of interest with more precision. The links are modeled as rigid bodies. The re-usable models of the rigid body and the 1 DOF actuated joints are used as building blocks for the complete model and will be presented in the next section. In Figure A.1 the joint locations and positive directions are shown. As can be seen in this figure each rigid body is described using multiple frames and the initial frame ψ_0 is located in the corner of the base. Each rigid body has three coordinate frames; one in the joint with the previous link called ψ^i, one at the center of mass of that link, which is ψ^k and one at the joint with the next link ψ^j. All are oriented in the same way as the inertial frame in the initial position. The initial position of the manipulator is folded with the end-effector next to the base.

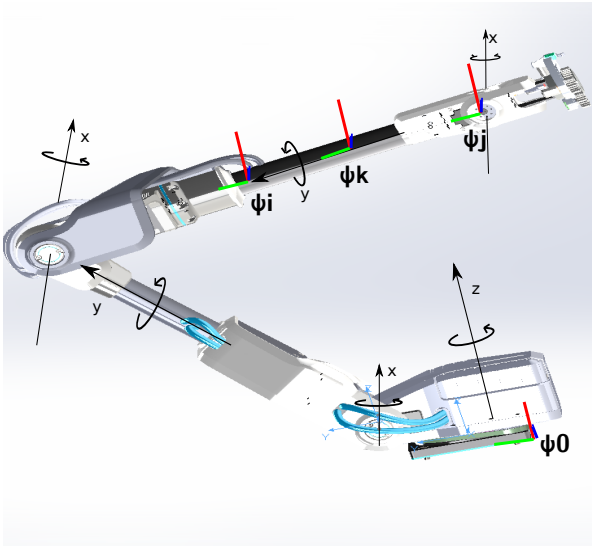

Figure A.1: Example of placement of frames for the link 5.

Rigid body submodel

The following notation will be used in this section:
$W^T = \begin{pmatrix}\tau & F\end{pmatrix}^T$: *wrench*
$T = \begin{pmatrix}\omega \\ v\end{pmatrix}$: *twist*
$I = \begin{pmatrix}J & 0 \\ 0 & M\end{pmatrix}$: *Inertia tensor in the principle inertia frame*
Where F: Force, τ: torque, ω: angular velocity, v: linear velocity, M: Mass matrix and J: Inertia matrix.

The model of the rigid-body dynamics "contains" inertia and gyroscopic effects, an externally applied wrench and a gravitational force. The wrench balance expressed in ψ^k can be written as [2]:

$$I^k \dot{T}_i^{k,0} = \begin{pmatrix}\tilde{\omega}_k^{k,0} & -\tilde{v}_k^{k,0} \\ 0 & -\tilde{\omega}_k^{k,0}\end{pmatrix} I^k T_k^{k,0} + (W^k)^T + G(q) \tag{A.1}$$

The left hand side of the equation (A.1) represents the component of inertia. The first component on the right hand side of the equation represents the gyroscopic and coriolis effects and the second component on the right represents the externally applied wrench. The externally applied wrenches include the joint reactions. $G(q)$ represents the gravitational forces that act upon the link.

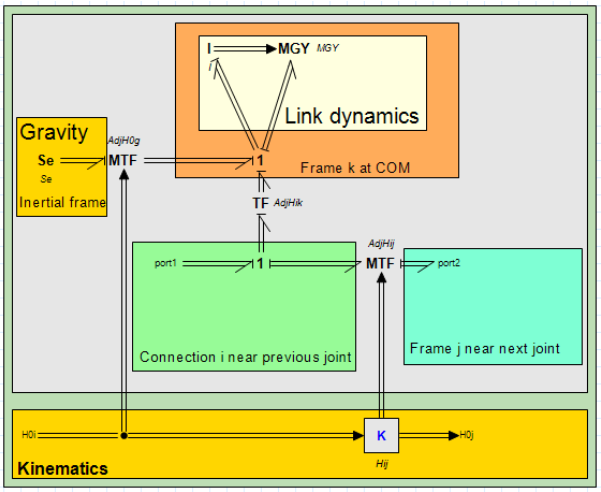

Figure A.2: Model of generalized rigid-body

The model of a generalized rigid-body is shown in

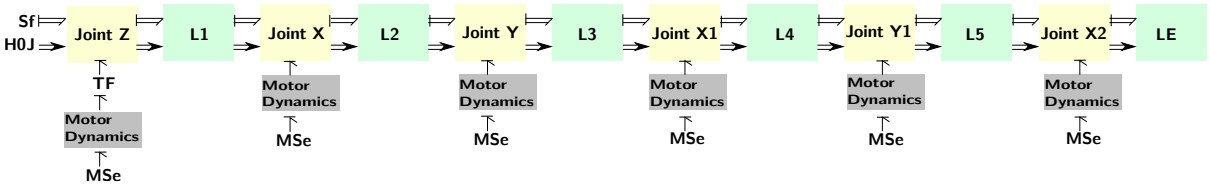

Figure A.3: Complete Model of the RoboShip manipulator

Figure A.2. The top 1-junction represents equation (A.1). The I-element represents the inertia and the MGY represents the gyroscopic element. The Se element represents the gravitational force that is applied on the rigid-body in the inertial frame and together with the wrench from the TF element this is the externally applied wrench. Port1 represents the connection with the previous joint and port2 represents the connection point with the next joint. The MTF between these ports is the transformation to transform the twists and wrenches from ψ^i to ψ^j. The TF element between the two 1-junctions repents the transformation matrix from ψ^i to ψ^k. In Figure A.2 the kinematic relation is also shown between the input H_i^0 and the output H_j^0. These matrices represent the configuration of the previous and next joints respectively with respect to the inertial frame. To go from H_i^0 to H_j^0 the following formula is used:

$$H_j^0 = H_i^0 \; H_j^i \tag{A.2}$$

A.2 Joint submodel

In this joint model it was chosen to only model the transfer of energy. The joint establishes an energetic connection between two links, it imposes a relation between wrenches and the twists of two bodies. The manipulator has six joints which are all 1 DOF actuated rotational joints. The wrench applied on the joint will result in the same wrench applied on both connected links which is represented with a 0-junction.
In Figure A.4 the generalized model of the joint is shown. In this model the effort port represents the torque generated by the motor and the R represents the friction in the degree of freedom. The TF element represents the conversion from joint torque to wrench and from twist to joint velocity. Since the joint actuator is related to one of the wrench components and the joint is constrained in all other directions, motion constraints need to be added. This is done by using high friction and low compliance connected to the joint. The MTF-element uses the twist generated by the motor to change coordinate frames of the wrench and twist from port1 to port2. This computed homogeneous matrix is then used to update the kinematics from ψ^j to ψ^i using the following relation:

$$H_i^0 = H_j^0 \; H_i^j \tag{A.3}$$

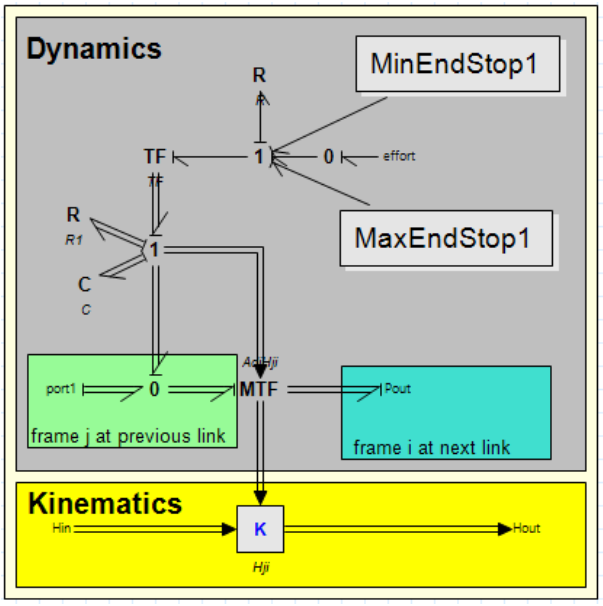

Figure A.4: Model of a generalized joint

A.3 Complete model of the manipulator

The complete model can be seen in Figure A.3. In this model all the links and joints have been coupled with both bonds (for the dynamics) and signals (for the kinematics). The Sf-element is a source of flow of zero specifying that the joint at that side is connected with the fixed world. The H_j^0 element specifies where the manipulator is located in the tank frame. The MSe elements are input torques that are computed by a controller.

References

[1] Dresscher, Douwe, et al. "Modeling of the youBot in a serial link structure using twists

and wrenches in a bond graph." (2010): 385-400.

[2] Stramigioli, Stefano & Herman Bruyninckx. "Geometry and screw theory for robotics." Tutorial during ICRA 2001 (2001).

B Kinematics verification using OptiTrack

Forward kinematics is used to determine the position and orientation error of the end-effector. To validate that the forward kinematics of the manipulator is correct, OptiTrack from NaturalPoint [1] (version 2.2.3) is used. The link lengths of the manipulator are obtained from the SolidWorks model which was used to manufacture the manipulator. For this reason the only unknown in the system is the actuator offset. The actuator offset is the measured encoder value when the joints angle is zero.

The OptiTrack system works with highly reflective spheres as markers. The markers have a diameter of 16 mm. To indicate a rigid body at least three markers have to be used to estimate the pose.

B.1 Method

Two sets off markers are used to determine the pose of the end-effector with respect to the origin of the manipulator. The first set is placed between the rails and the base of the manipulator. This set is used to indicate the origin of the manipulator and is fixed. The second set of markers consists of four markers and is mounted right after joint 6. The end-effector marker set is shown in Figure B.1. For the end-effector it was chosen to use four markers instead of three, such that the pose can still be determined if one marker is undetectable. The pivot point of a rigid body is the point of which the location is tracked by the OptiTrack system. The pivot point of the rigid body was chosen at one of the markers, because the position of this marker is always known with respect to joint 6.

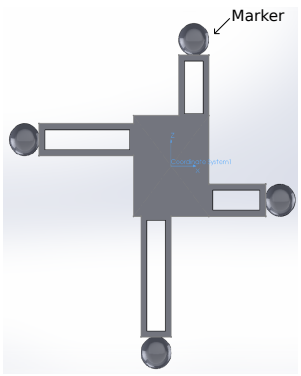

Figure B.1: End-effector marker set

The location of the end-effector is determined with OptiTrack. This location is compared to the position of the end-effector that is obtained by using forward kinematics.

The angle of a joint is determined by:

$$q(i) = \frac{(\text{encoder output} - \text{actuator offset}) * 360^\circ}{\text{scale}} \tag{B.1}$$

The scale in this equation is the total number of pulses per revolution of the encoder. The encoders in the Dynamixel MX-106 and MX-28 both have 4096 pulses per revolution which corresponds to a resolution of 0.088° To determine the correct actuator offset, different actuator offsets are used to calculate the angle. With forward kinematics the position of the end-effector is then calculated and compared with the position that was obtained with OptiTrack. A cost-function is used to determine the best actuator offset for each joint. The cost is calculated by:

$$C = \sqrt{x_e^2 + y_e^2 + z_e^2} \tag{B.2}$$

Where x_e^2, y_e^2 and z_e^2 are the sums of the absolute errors in the x-,y- and z-direction over a trajectory respectively. The actuator offsets are the values for which the cost is minimal.

Since the manipulator is 6 DOF a position in space can be reached by multiple configurations. When a trajectory is followed with the end-effector an optimal set of actuator offsets that minimizes the error between the measured end-effector pose and the predicted end-effector pose can be found. The trajectory is made manually such that the controller will not influence the measurements.

B.2 Optimized actuator offsets

The initial guess, obtained by visual inspection, of actuator offsets and the optimized actuator offseta are used to calculate the position of the end-effector along the trajectory, see Figure B.2. Both are compared with the position trajectory of the end-effector measured with the OptiTrack system. The error between those are shown in Figure B.3. The data when the end-effector was not tracked properly is replaced by NaNs in the error calculation and is thus not shown in the plots. In this figure it is visible that the optimized actuator offsets do not result in an error equal to zero. The error of both the initial guess and that of the optimized motor offset are larger then expected. A maximum error of 1 cm was expected since larger errors can be measured manually. If every motor offset would be 1 count off in the same direction

the position error would be 0.13 mm, 0.11 mm and -1.40 mm in the x-, y- and z-direction respectively. The error of 10 cm is much more than expected.

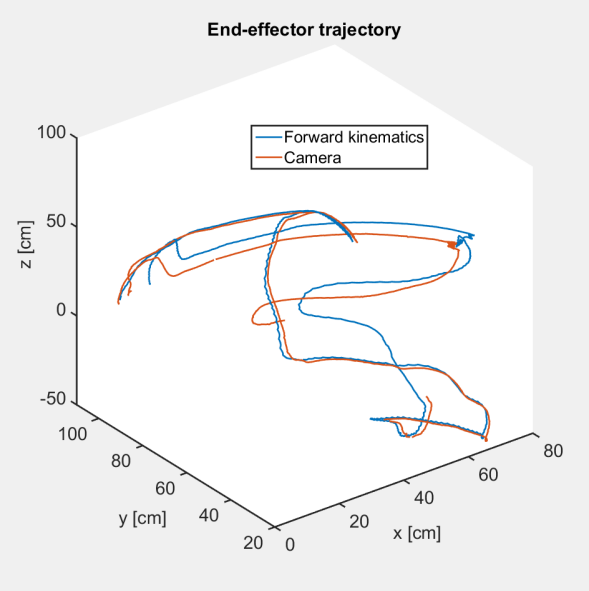

Figure B.2: Trajectory of the end-effector.

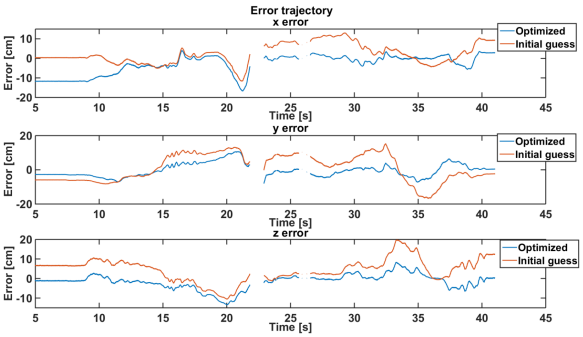

Figure B.3: Position error in each direction

If the forward kinematics are correct and the homogeneous matrix from joint 6 to the pivot point would be wrong, it would be expected that the norm of the error is constant. As can be seen in Figure B.4 this is not the case. The error is calculated between the position measured with OptiTrack and forward kinematics using the optimized actuator offset.

B.3 Area calculation

To validate that the OptiTrack system is able to determine the position of the markers, an experiment was performed. In this experiment the end-effector

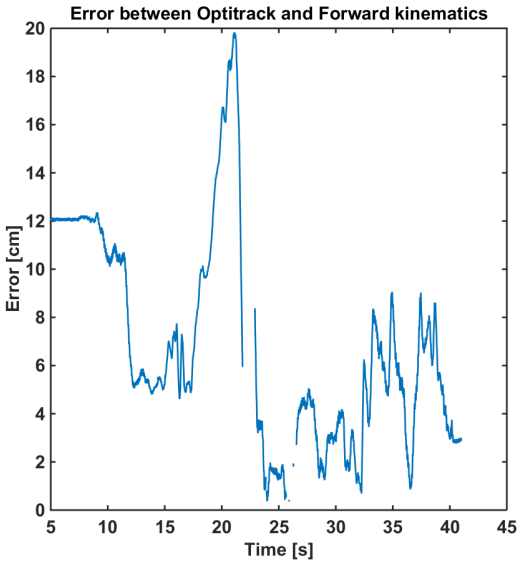

Figure B.4: Position error in each direction

Triangle	Area $[cm^2]$
Triangle 1	45.5
Triangle 2	50.9
Triangle 3	61.6
Triangle 4	67.0

Table B.1: Area of the triangles

is moved in the OptiTrack room and all four markers are tracked. It is possible to create four triangles between the markers. The markers are all attached to one rigid body and do not move relative to each other. Therefore, the area of these triangles is constant. The triangle is specified by vectors u and v that originate at one vertex. The area can then be calculated by [2]:

$$A = \frac{1}{2} \mid u \times v \mid \tag{B.3}$$

The error between the calculated area of the triangles and the calculated area from OptiTrack during the experiment is given in Figure B.5.
The total area of the triangles can be seen in Table B.1
The error in the area of the triangles is not constant which was not expected if the marker positions are correctly calculated by OptiTrack.

B.4 Discussion

From the experiments shown in the previous sections it is not possible to conclude that the forward

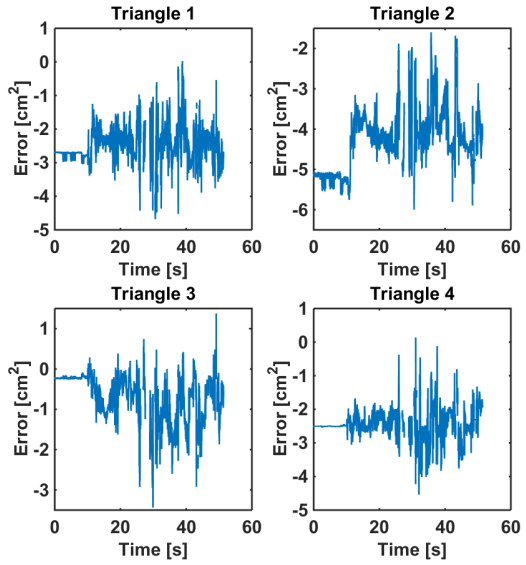

Figure B.5: Error in the area of the triangles

kinematics is incorrect or if the OptiTrack system is not able to measure the position of the end-effector correctly. More experiments need to be done to identify the exact cause.

References

[1] OptiTrack, Natural Point. `http://www.optitrack.com/`

[2] Triangle Area- WolframAlpha. `http://mathworld.wolfram.com/TriangleArea.html`

C Obstacle avoidance

When the manipulator is controlled to a point of interest, it is of importance that the manipulator and tank are not damaged by colliding with objects. This challenge can be overcome by using an obstacle avoidance approach. It must be able to reach the point of interest, which is located on a wall which is an object as well. The obstacles in this scenario are the manipulator itself, the rails, the walls, the piping and the robot on which the manipulator is mounted.

C.1 Recommended approach

Generally, the obstacle avoidance problem can be solved with two classes of strategies: global (planning) or local (control) [1]. The global option guarantees to find a collision-free path from an initial point to a goal, if a path exists.

The global option often operates in the configuration space into which all obstacles are mapped. A collision free path can be found in the free space of the configuration space [2]. This approach is computationally demanding and usually does not rely on sensor feedback is are therefore only suitable for well-defined static environments.

Local approaches treat obstacle avoidance as a control problem and can use information from sensors which makes the approach more flexible. The drawback of this approach is that it can find a global suboptimal path and can get stuck in local minima [3].

For this project, where the configuration is not important, it is best to use a local planner to include moving obstacles. To avoid obstacles the manipulator has to move away from the obstacles. This can be done by calculating the minimal distance of the manipulator to an obstacle and applying a force in the opposite direction to the manipulator if the distance is smaller than the critical distance.

C.2 Modeling

The environment of one compartment in a BWT consists of six walls, two openings and a rail. The minimal distance needs to be calculated between objects that can collide with each other. In Table C.1 can be seen which part of the manipulator can collide with which object. Link 1 is the rotating base.

Link	Can hit	Can be hit by
Link1	-	End-effector + Link 5
Link2	Rails + Robot	-
Link3	Rails + Robot + Wall + Piping	-
Link 4	Rails + Robot + Wall + Piping	-
Link 5	Rails + Robot + Wall + Piping	-
End-effector	Rails + Robot + Wall +Piping	-

Table C.1: Possible collisions

To calculate the minimal distance between objects, a simplified geometrical representation of each object can be used. The rails and the links are modeled as finite line segments with a large enough safety distance, such that the whole object is contained in a cylinder. The walls are modeled as rectangles with a safety distance such that the manipulator will not touch the walls. For the end-effector the wall on which the point of interest is located, is also an obstacle. The wall can be divided in more than one rectangle to leave an open space for the end-effector to overcome this problem. The wall could also be modeled such that if the end-effector is far away the wall is an obstacle but as it nears the point of interest, the wall is not an obstacle anymore.

C.3 Minimal distance calculation

Distance between line segments

To calculate the minimal distance between two line segments the algorithm given by David Eberly [4] is used. This algorithm applies to segments in any dimension. The endpoints of the first segment are denoted by P_0 and P_1 and of the second segment Q_0 and Q_1. The segments can be parametrized as $P(s) = (1-s)P_0 + sP_1$ and $Q(t) = (1-t)Q_0 + tQ_1$. The squared distance between two points on the line segments is the quadratic function:

$$R(s,t) = |P(s) - Q(t)|^2 \tag{C.1}$$

To calculate the minimal distance the function R(s,t) should be minimized over the unit square $[0,1]^2$.

Point to rectangle

A link of the manipulator can only collide with the wall at the end of the line segments. Therefore,

only the distance between the endpoints and the rectangle have to be calculated. This can be done by calculating the minimal distance between a point and a plane. However, the plane is infinite and a wall is not. When the critical location is in region 0 (see Figure C.1, region 0 represents the square wall) the minimal distance is simply the distance between the plane and the point. If the critical point is in region 2, 4, 6 or 8 then the problem can be rewritten as a calculation for the distance between a line segment and a point. If the critical point is in region 1, 3, 5 or 7 then the minimum distance is between a point and a point.

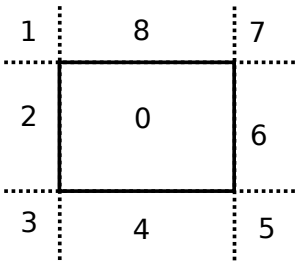

Figure C.1: Distance from point to rectangle

C.4 Foreseen problems and possibilities

As stated before, the risk of a local approach is that no optimal path is found and the manipulator stays in a local minimum. This can happen quite easily in this case, for example, when the setpoint is below the rails, but since the obstacle avoidance is in place the manipulator takes the approach to take the path above the rails. It is possible that the manipulator is not able to reach the point of interest anymore. To overcome this issue a path planner could be considered as an appropriate solution which generates setpoints for the end-effector. This way a trajectory is generated such that the manipulator is forced to take a different route underneath the rails.

The method described above assumes known obstacles with known positions. However, not every BWT is the same and mapping the environment by describing each object with an geometrical shape, is very time consuming. An option is to think about exploring the possibilities of SLAM to create a map of the environment and obtain the obstacles from that map. An other option is to mount sensors on the manipulator that can measure the distance between objects and when it passes a certain threshold a force is applied to the manipulator at the critical point.

References

[1] Kim, J. O., & Khosla, P. K. (1992). Real-time obstacle avoidance using harmonic potential functions. Robotics and Automation, IEEE Transactions on, 8(3), 338-349.

[2] Lozano-Perez, Tomas. "Spatial planning: A configuration space approach." Computers, IEEE Transactions on 100.2 (1983): 108-120.

[3] ŽlajpahĀ, Leon and PetriĀ, Tadej, Obstacle Avoidance for Redundant Manipulators as Control Problem

[4] Eberly, David, Robust Computation of Distance Between Line Segments