UNIVERSITY OF AGDER

Automatic Path Planning for Door-Milling Application Using an Industrial Robot

Kai Egil Berntsen
André Bleie Bertheussen

Supervisors
Erlend Eltervåg
Ilya Tyapin

This master's thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education. However, this does not imply that the University answers for the methods that are used or the conclusions that are drawn.

University of Agder, 2018
Faculty of Engineering and Science
Department of Engineering Sciences

Abstract

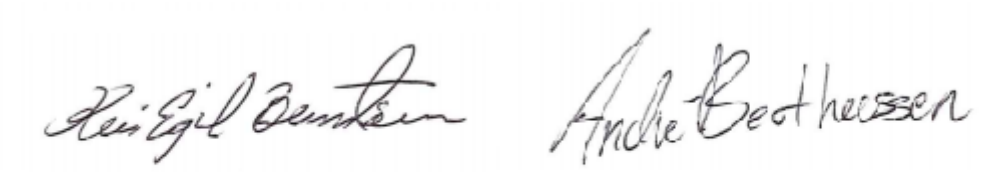
This report covers automatic path planning for a 6 degree of freedom industrial robot used in a milling application. The kinematics are derived for the ABB IRB6600 robot used in this master thesis. Different sensors for determining the work-object pose in relation to the robot are evaluated, and a suggested vision system is presented and tested. When industrial robots are used in high precision contact applications, such as milling, the robot stiffness must be considered. Therefore, a joint stiffness analysis is conducted for the ABB IRB6600. The results from the stiffness analysis are, in addition to the milling force data, used to compensate for deflection when generating the robot path trajectory. The same data is also used to find favourable operation configurations for the robot. Important elements when using an industrial robot in a milling application are presented and evaluated to assure a good end-product. Finally, a method for automatic generation of robot path trajectory, extracted from a database, containing door data is presented. Here, the previously mentioned sensor system is used in addition to the path generation code to create a *.txt file, which can be read by RAPID code. The RAPID code is further executed by the robot to mill the desired features in the door.

Preface

This master thesis has included both practical and theoretical work, and it has given valuable experience in combining multiple engineering disciplines. We would like to show gratitude towards our supervisor, Ilya Tyapin at the University of Agder, who has given us valuable help and guidance throughout this master thesis.

We would also like to thank Robot Norge AS and Nordicdoor AS for giving us the opportunity to work with this exciting project. Further, we thank Geir Hovland for helping us with reconfiguration of the robot cabinet, and installation of the milling spindle.

Grimstad, May 25, 2018



Kai Egil Berntsen André Bleie Bertheussen

Contents

List of Figures

1 Introduction

In this chapter the problem statement and motivation for this thesis are presented. Assumptions and limitations set for this thesis are also given, together with the requirements. This chapter also includes a short description of the report structure, together with a description of the work method.

1.1 Motivation and Problem Statement

Traditionally, the use of industrial robots has mainly been limited to large production series in large companies, where they have been handling standardized products in great numbers. In order to meet demands from the industry, and expand the use of robots in smaller companies, development of flexible solutions where the robots can perform multiple tasks, and handle different items without human intervention is of interest.

One important element in this development is the identification and definition of the robot work object. This can either be achieved through the use of advanced sensor and vision systems, or through digitization of CAD files into databases which can be utilized by the robot, or both. Another element of great interest is how the robot application can automatically generate its own program containing both path trajectory and desired actions to meet the production requirement, based on input from the identified work object.

The demand for automation in smaller production lines, due to cost reduction, is increasing. RobotNorge AS considers the described issues to be of great importance for future projects. An ongoing project is to develop a solution where robots can replace an eleven-axis CNC machine, in the production line at the door manufacturer Nordic Door AS. Nordic Door deliver doors for a wide variety of buildings, like hotels, universities and hospitals. The doors are delivered in a great variety of sizes, design and production numbers. Hence, Nordic Door want a versatile, accurate and reliable solution for their production line.

This master thesis is based on the following problem statement: For an industrial robot door-milling application, the path trajectory is to be automatically generated from provided door data, in order to mill desired features in a wide variety of wooden doors. Therefore, this thesis presents a method for utilizing door data to generate the robot path trajectory and milling actions, adapted to each door configuration. In order to achieve good results, it is of importance to know the position of the work object with high accuracy. Therefore, several different sensor solutions are elaborated, and a suggested vision system able to determine the work object position and orientation is presented.

When using industrial robots for high precision contact applications, such as milling, deflection due to exerted forces has to be considered. This thesis presents a method for analysing the joint stiffness of a 6-DOF industrial robot, and how these results are used to position the work

object related to the robot, and also to correct the generated path, in order to achieve better end results.

1.2 Assumptions and Limitations

Robotic cell designed by RobotNorge is shown in figure 1.1. This is part of a suggested solution to a door milling application at the Nordic Door factory. The suggested solution consists of one handling robot (H1), a carousel (K1) to transport the doors from handling robot to milling area, and two milling robots (F1,F2), which will mill one side of the door each.

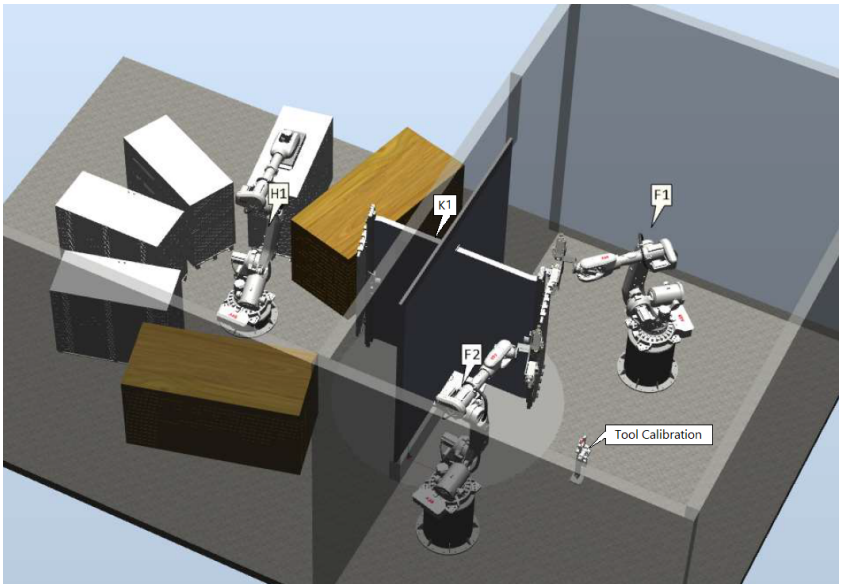


Figure 1.1: Robot Cell [Appendix A.1]

When visiting Nordic Door, it became clear that the handling robot and carousel shown in figure 1.1, may not be the optimal solution. Other possible solutions for the transport between handling robot and milling cell were discussed. However, the transport part of the application is considered out of the scope for this master thesis. Hence, the doors are assumed to be securely presented in front of the milling robots. There could be deviations in the placement of the doors, but after discussing with Nordic Door, it is assumed that the position and orientation are locked in three degrees of freedom (3-DOF).

Further, the focus has been directed towards one of the two milling robots, since only one robot was available for this project. Further, there are restrictions related to bringing lab equipment such as a Faro laser tracker, out of the lab. The idea is also that the solution for one milling robot can, with small adjustments, be adapted to a second one.

The different parts of the application, such as the vision system for determining exact door pose (position and orientation), automatic robot trajectory generation, and path correction based on a robot stiffness analysis, are all tested individually.

1.3 Requirements

- Based on door features stored in a database, the application has to generate the robot path trajectory, and desired actions.
- Both cost and cycle time has to be considered in development of the application.
- The application must be able to handle doors in different sizes.
  Height: 1000-3000 mm, Width: 190-1350 mm, Thickness: 30-100mm.
- Tolerances in milled features are set to 0.5 mm.

1.4 Work Method

In order to work efficiently in a project, planning and communication is essential. A good project management strategy ensures that the members of the group contribute in an efficient way. In the beginning of this project, a Gantt chart was made using Microsoft Visio. The Gantt chart is divided into main parts of the projects. Further, a time line is made, with a start and a deadline. The Gantt chart can be shown in figures 1.2 - 1.5.

Gantt Chart

ID	Task Name	Start	Finish	Duration
1	Defining the Master thesis	03.01.2018	26.01.2018	18d
2	Info gathering	03.01.2018	26.01.2018	18d
3	Task: Thesis, Gantt, Description	10.01.2018	01.02.2018	17d
4	Limitations	30.01.2018	06.02.2018	5d
5	Kinematics	01.02.2018	19.02.2018	13d
6	Trajectory Generation (Path planning)	12.02.2018	08.03.2018	18d
7	Sensors (Probing, camera, lidar, laser, etc)	05.03.2018	30.03.2018	20d
8	Robot Stiffness Analysis	28.03.2018	12.04.2018	12d
9	Robot cell optimization (Optimal robot position)	12.04.2018	24.04.2018	8d
10	Process Force Correction	23.04.2018	08.05.2018	12d
11	Safety	07.05.2018	09.05.2018	3d
12	Documentation / Report writing	10.01.2018	25.05.2018	98d
13	Proof reading	18.05.2018	25.05.2018	6d

Figure 1.2: Gantt Chart Section.1

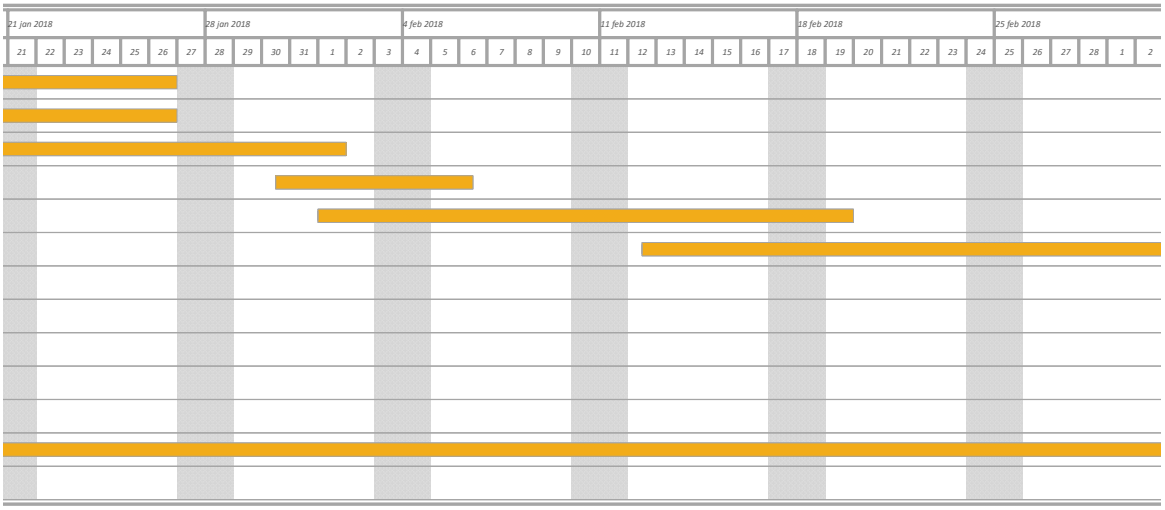


Figure 1.3: Gantt Chart Section.2

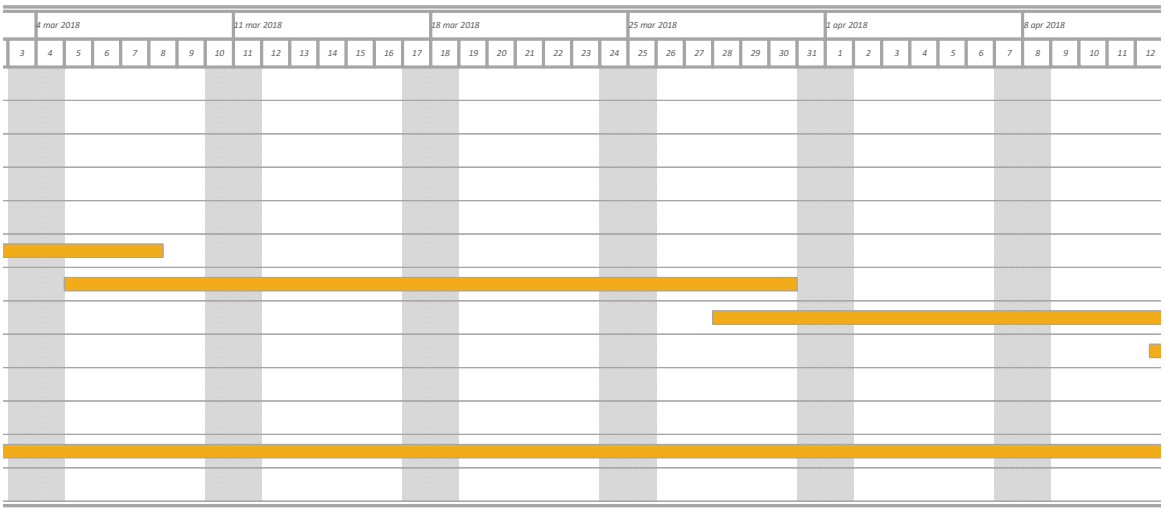


Figure 1.4: Gantt Chart Section.3

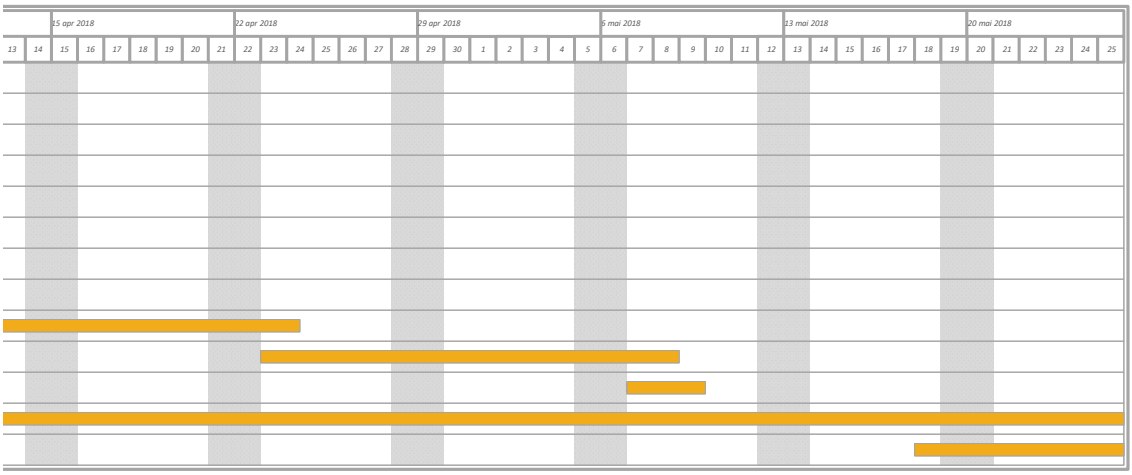


Figure 1.5: Gantt Chart Section.4

According to the Gantt chart, the definition of the thesis was the foremost priority. Here, we travelled to RobotNorge's headquarters at Klepp, Stavanger to fully define the task in co-operation with external supervisor. Further, we gathered information of the thesis content. Limitations of the thesis were later discussed with both internal and external supervisor. With the limitations determined, the kinematics was derived simultaneously with the path trajectory generation and sensor evaluation. Further, the stiffness analysis was conducted. Using the derived kinematics together with stiffness data; process force correction and robot cell optimization could be determined.

Throughout this master thesis, several different softwares have been used, such as: MATLAB®, NI LabVIEW, ABB Robot Studio, ESI SimulationX and Maplesoft Maple. This report is written using LaTeX.

1.5 Report Structure

This section gives a short description of each chapter in this master thesis. A short summary of each chapter is listed below.

- Chapter 1 - **Introduction**
  In the introduction chapter, the problem statement and motivation are presented. The assumptions are made, and the limitations set for this thesis are also listed, together with the given requirements. An overview of the report structure is also given in this chapter. Finally, the work method is described.
- Chapter 2 - **Theory**
  This chapter contains theory and equations regarding robot kinematics, including the Jacobian matrix. Theory vital to the choice of sensors are presented followed by an introduction to robot stiffness and kinematic performance. An overview of different path correction methods are also included in this chapter together with an introduction to milling.
- Chapter 3 - **Method**
  The different methods proposed and used in this master thesis are presented in this chapter. First a stiffness identification method is presented, followed by an evaluation of the influence of pre loading the joints in the stiffness analysis. The influence of link stiffness compared to joint stiffness is also included. A simulation model of the robot is presented in SimulationX software, which is used to evaluate the results of the stiffness analysis. Selecting a region of operation based on different door sizes, robot kinematics and kinematic performance is described, followed by a description of methods for work-object localization with several different sensors. Further, this chapter describes how door specifications listed in a database can be used to generate a robot path trajectory and desired milling action.
- Chapter 4 - **Results**
  In this chapter, the achieved results are presented. The results from the stiffness analysis, together with comparison to the simulation model are presented, followed by the results from the evaluation of pre-loading influence, link stiffness and kinematic performance. The resulting region of operation is presented, and finally the results of the path generation and milling sequence are presented.
- Chapter 5 - **Discussion**
  The results and the methods used to achieve them are elaborated in the discussion chapter, together with suggestions for further work.
- Chapter 6 - **Conclusion**
  The conclusions drawn in this thesis are presented in the conclusion chapter.

2 | Theory

This chapter contains theory and equations regarding robot kinematics, including the Jacobian matrix. Theory vital to the choice of sensors and how they can be used are presented followed by an introduction to robot stiffness and kinematic performance. An overview of different path correction methods is also included in this chapter together with an introduction to milling.

2.1 Kinematics

The forward kinematics determine the pose of the end-effector, or tool centre point (TCP), based on the robot joint values, while the inverse kinematics determine the robot joint values from the end-effector pose [2].

Figure 2.1 shows the dimensions of the ABB IRB6600/175-2.55 robot used for experiments and tests throughout this report. The robot is shown in home position, where all the joint angles are 0°. All the 6 joint axes, with positive and negative directions of rotations are shown in figure 2.2.

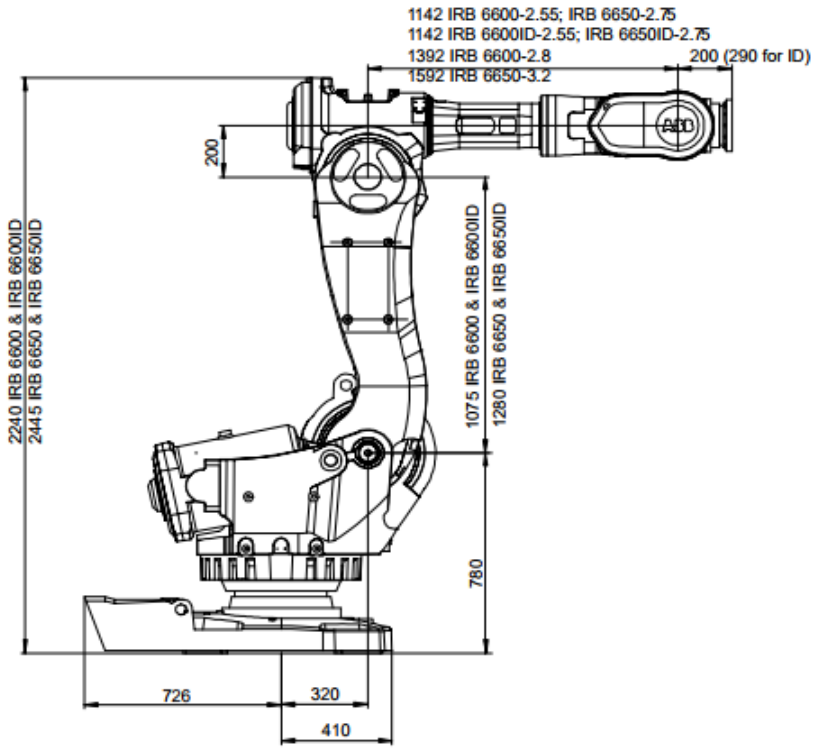


Figure 2.1: Illustration of the IRB6600/175-2.55 [1]

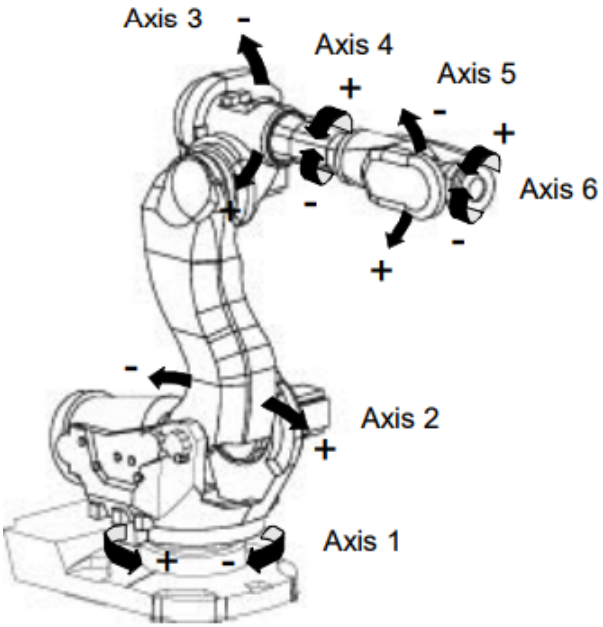


Figure 2.2: The Six Axes of the IRB6600/175-2.55 [1]

2.1.1 Forward Kinematics

The forward kinematics for the IRB6600 robot is derived using the Denavit-Hartenberg convention. The method for a 6-DOF industrial robot, like the IRB6600, is first to define a reference frame at each joint(q), where the z-axis is the axis of rotation, and the direction of rotation corresponds to the actual robot, like shown in figure 2.2.

The next step is then to describe the connection between each joint by the help of the four parameters, in the given order: Rotation around z-axis, translation along z-axis, translation along x-axis and rotation around x-axis, in this order. For a 6-DOF robot, the DH-table will have 6 rows, where each row represents the relation between two joints. The DH-table for the IRB6600 in home position is shown in table 2.1.

Table 2.1: Denavit-Hartenberg convention

	$\boldsymbol{R}_z$ [rad]	$\boldsymbol{T}_z$ [m]	$\boldsymbol{T}_x$ [m]	$\boldsymbol{R}_x$ [rad]
$\boldsymbol{A}_1$	q_1	0.780	0.320	$-\frac{\pi}{2}$
$\boldsymbol{A}_2$	$q_2 - \frac{\pi}{2}$	0	1.075	0
$\boldsymbol{A}_3$	q_3	0	0.200	$-\frac{\pi}{2}$
$\boldsymbol{A}_4$	q_4	1.142	0	$\frac{\pi}{2}$
$\boldsymbol{A}_5$	q_5	0	0	$-\frac{\pi}{2}$
$\boldsymbol{A}_6$	$q_6 + \pi$	0.200	0	0

Each homogeneous transformation matrix is calculated individually:

$$\begin{aligned}
\boldsymbol{A}_1 &= [\boldsymbol{R}_z(q_1)] \cdot [\boldsymbol{T}_z(0.780)] \cdot [\boldsymbol{T}_x(0.320)] \cdot [\boldsymbol{R}_x(-\frac{\pi}{2})] \\
\boldsymbol{A}_2 &= [\boldsymbol{R}_z(q_2 - \frac{\pi}{2})] \cdot [\boldsymbol{T}_z(0)] \cdot [\boldsymbol{T}_x(1.075)] \cdot [\boldsymbol{R}_x(0)] \\
\boldsymbol{A}_3 &= [\boldsymbol{R}_z(q_3)] \cdot [\boldsymbol{T}_z(0)] \cdot [\boldsymbol{T}_x(0.200)] \cdot [\boldsymbol{R}_x(-\frac{\pi}{2})] \\
\boldsymbol{A}_4 &= [\boldsymbol{R}_z(q_4)] \cdot [\boldsymbol{T}_z(1.142)] \cdot [\boldsymbol{T}_x(0)] \cdot [\boldsymbol{R}_x(\frac{\pi}{2})] \\
\boldsymbol{A}_5 &= [\boldsymbol{R}_z(q_5)] \cdot [\boldsymbol{T}_z(0.200)] \cdot [\boldsymbol{T}_x(0)] \cdot [\boldsymbol{R}_x(-\frac{\pi}{2})] \\
\boldsymbol{A}_6 &= [\boldsymbol{R}_z(q_6 + \pi)] \cdot [\boldsymbol{T}_z(0.200)] \cdot [\boldsymbol{T}_x(0)] \cdot [\boldsymbol{R}_x(0)]
\end{aligned}$$

Finally, the position and orientation of the end effector related to the initial coordinate frame is given by $\boldsymbol{T}_6^0$.

$$\boldsymbol{T}_6^0 = \boldsymbol{A}_1 \cdot \boldsymbol{A}_2 \cdot \boldsymbol{A}_3 \cdot \boldsymbol{A}_4 \cdot \boldsymbol{A}_5 \cdot \boldsymbol{A}_6 \tag{2.1}$$

where:

$$\boldsymbol{R_z} = \begin{bmatrix} \cos(q) & -\sin(q) & 0 & 0 \\ \sin(q) & \cos(q) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$\boldsymbol{R_x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(q) & -\sin(q) & 0 \\ 0 & \sin(q) & \cos(q) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

$$\boldsymbol{T_z} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$\boldsymbol{T_x} = \begin{bmatrix} 1 & 0 & 0 & L \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

The MATLAB®script used to calculate the forward kinematics are given in the appendix A.2.

2.1.2 Inverse Kinematics

The inverse kinematics is used to calculate the joint angles required to place the end-effector in a desired pose. While the forward kinematics always has one unique solution, this is not the case for the inverse kinematics. Not all end-effector poses are possible to achieve, hence there is no solution to the inverse kinematics for these end-effector poses. Also if there is a solution to the inverse kinematic, this solution may not be unique. Figure 2.3 show commonly used names on robot parts, for describing different configurations. In cases where there are multiple solutions, they are determined by different robot configurations known as base upwards/backwards, shoulder left/right, elbow up/down and wrist up/down.

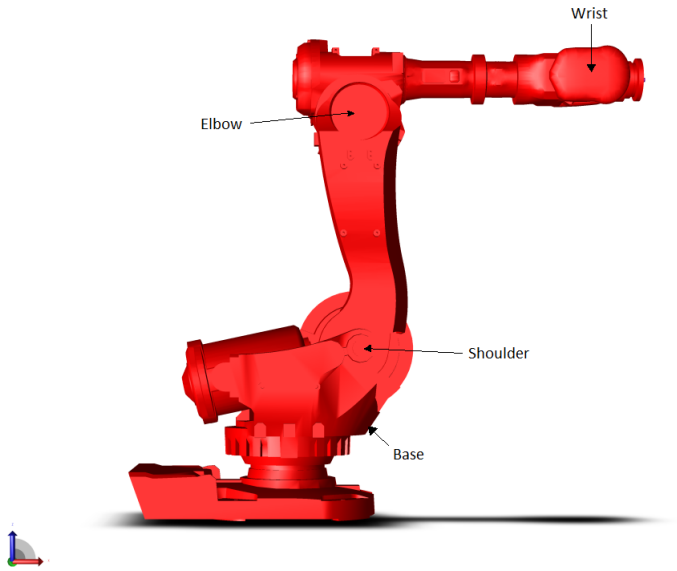


Figure 2.3: Side-view of the IRB 6600

General Inverse kinematics

As shown in [2], the general inverse kinematics problem can be stated as shown in equation 2.6. The desired pose of the end-effector is given by a 4x4 homogeneous transformation matrix $\boldsymbol{H}$, where $\boldsymbol{R}$ is a 3x3 rotational matrix, and $\boldsymbol{o}$ is a 3x1 position vector.

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{o} \\ 0 & 1 \end{bmatrix} \in SE(3) \tag{2.6}$$

With $\boldsymbol{R} \in So(3)$ find one or all solutions to the equation:

$$\boldsymbol{A}_1(q_1) \cdots \boldsymbol{A}_n(q_n) = \boldsymbol{T}_n^0(q_1, ..., q_n) = \boldsymbol{H} \tag{2.7}$$

The results of equation 2.7 are twelve non-linear equations with n unknown variables, which can be written as:

$$\boldsymbol{T}_{ij}(q_1, ..., q_n) = h_{ij}, \qquad i = 1, 2, 3, \qquad j = 1, .., 4 \tag{2.8}$$

where $\boldsymbol{T}_{ij}$ and h_{ij} refer to the twelve non trivial entries of $\boldsymbol{T}_n^0$ and $\boldsymbol{H}$, respectively. The bottom row of both $\boldsymbol{T}_n^0$ and $\boldsymbol{H}$ are (0,0,0,1), hence four of the sixteen equations represented by equation 2.7 are trivial.

Decoupling

Solving the inverse kinematics, and finding the equations to calculate joint values can be quite difficult on a robot with multiple degrees of freedom. However, investigating a selection of 6-DOF industrial robots, one common feature for many of these is that the three last joints intersect in one point. This is called a spherical wrist, and for these robots the inverse kinematics can be divided into two separate and much simpler calculations. This approach is called kinematic decoupling [2], or Pieper's solution, after Donald Lee Pieper [3].

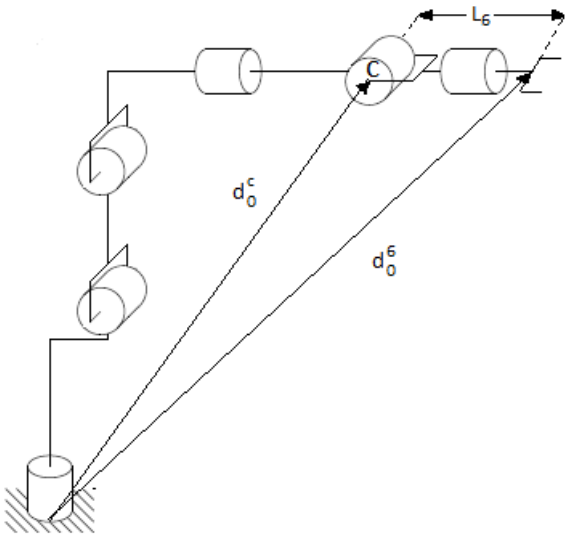


Figure 2.4: Decoupling IRB6600

From figure 2.4 it can be seen that only the three first joints have any effect on the wrist position $\boldsymbol{o}_c$. It can also be seen that since both position and orientation of the end-effector is given in $\boldsymbol{H}$, and the distance L_6, between c and end-effector is known from robot data sheet [1], $\boldsymbol{o}_c$ can be calculated:

$$\boldsymbol{o}_c = \boldsymbol{o} - L_6 \cdot \boldsymbol{R} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{2.9}$$

Now, the angles of the first three joints, q_1, q_2 and q_3 are determined based on the wrist coordinates given by equation 2.9. In figure 2.5, $\boldsymbol{o}_c$ is projected onto the $x_0 - y_0$ plane, and it can be seen that:

$$q_1 = tan^{-1}\left(\frac{y_c}{x_c}\right) \tag{2.10}$$

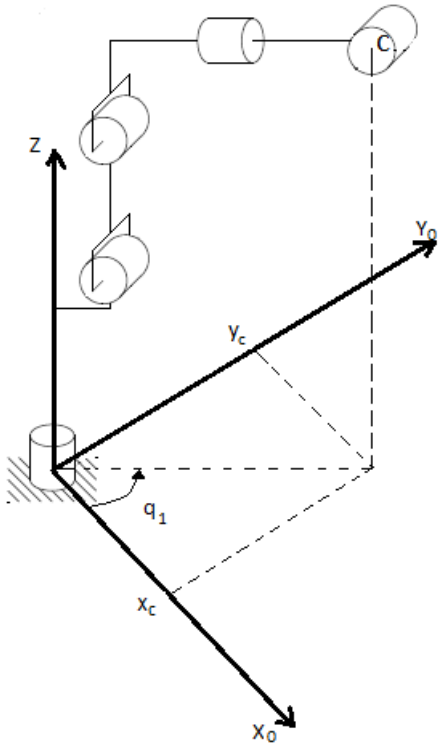


Figure 2.5: Finding q1

The next step is to calculate q_2 and $q3$. Figure 2.6 shows the simplified robot from joint 2 to joint 5 projected on to $x_0 - z_0$ plane.

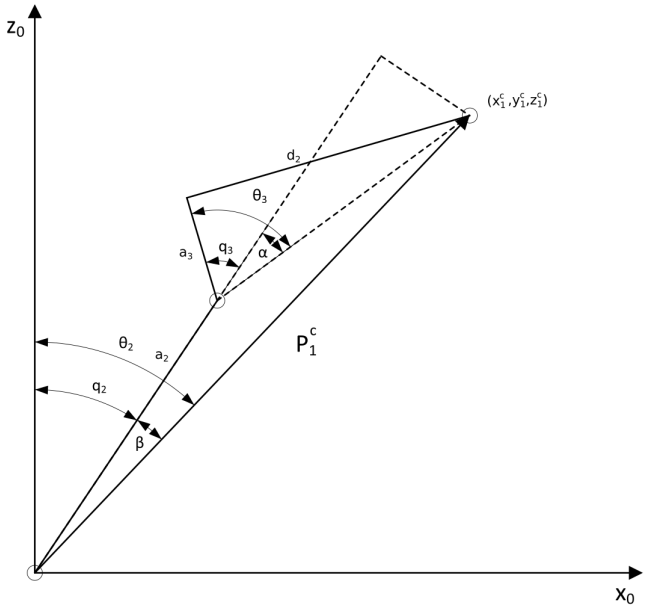


Figure 2.6: Projection of Robot from Joint 2 to Joint 5

In the projection of the robot shown in figure 2.6, the robot configuration is with elbow down. However, since α is the angle that is calculated and q_3 is just offset with $-\theta_3$, the calculated

configuration is actually elbow up, and α from equation 2.19 is therefore positive in the configuration shown in figure 2.6. If the calculated configuration should be elbow down, q_3 has to exceed θ_3. Keeping the latter in mind while examining the geometry it is found that:

$$q_2 = \theta_2 - \beta \tag{2.11}$$
$$q_3 = \alpha - \theta_3 \tag{2.12}$$

$$\theta_2 = tan^{-1}\left(\frac{\sqrt{x_1^{c2} + z_1^{c2}}}{-y_1^c}\right) \tag{2.13}$$

$$\boldsymbol{P}_1^c = \begin{bmatrix} x_1^c \\ y_1^c \\ z_1^c \\ 1 \end{bmatrix} = inv(\boldsymbol{T}_0^1) \cdot \begin{bmatrix} x_0^c \\ y_0^c \\ z_0^c \\ 1 \end{bmatrix} \tag{2.14}$$

$$\boldsymbol{T}_0^1 = \boldsymbol{A}_1(q1) = [\boldsymbol{R}_z(q_1)] \cdot [\boldsymbol{T}_z(0.780)] \cdot [\boldsymbol{T}_x(0.320)] \cdot [\boldsymbol{R}_x(-\frac{pi}{2})] \tag{2.15}$$

$$\beta = tan^{-1}\left(\frac{Li_3 \cdot sin(\alpha)}{Li_2 + Li_3 \cdot cos(\alpha)}\right) \tag{2.16}$$

$$Li_2 = a_2 \tag{2.17}$$

$$Li_3 = \sqrt{{a_3}^2 + {d_2}^2} \tag{2.18}$$

$$\alpha = tan^{-1}\left(\frac{\pm\sqrt{1 - D^2}}{D}\right), \qquad Elbow\ up/down \tag{2.19}$$

$$D = \frac{x_1^{c2} + y_1^{c2} + z_1^{c2} - Li_2^2 - Li_3^2}{2 \cdot Li_2 \cdot Li_3} \tag{2.20}$$

$$\theta_3 = \frac{\pi}{2} - tan^{-1}\left(\frac{a_3}{d_2}\right) \tag{2.21}$$

where:

$\alpha, \beta, \theta_2, \theta_3$	-	Calculation Help Angles.
x_1^c, y_1^c, z_1^c	-	Wrist Position Related to Robot Origin.
x_0^c, y_0^c, z_0^c	-	Wrist Position Related to Joint 2.
a_2	-	Length of Link between Joint 2 and Joint 3.
a_3	-	Length of Link between Joint 3 and Joint 4.
d_2	-	Length of Link between Joint 4 and Joint 5.

Now that the three first joint variables are found, it is possible to determine the orientation transformation matrix for the first three joints $\boldsymbol{R}_3^0$. This makes it possible to compute the orientation of the end-effector relative to $\boldsymbol{o}_c$.

$$\boldsymbol{R} = \boldsymbol{R}_3^0 \cdot \boldsymbol{R}_6^3 \tag{2.22}$$

This yields that

$$\boldsymbol{R}_6^3 = (\boldsymbol{R}_3^0)^{-1} \cdot \boldsymbol{R} \tag{2.23}$$

From the forward kinematics we can compute the transformation matrix $\boldsymbol{T}_6^3$

$$\boldsymbol{T}_6^3 = \boldsymbol{A}_4 \cdot \boldsymbol{A}_5 \cdot \boldsymbol{A}_6$$

Then the orientation transformation matrix $\boldsymbol{R}_6^3$ is

$$\boldsymbol{R}_6^3 = \begin{bmatrix} s\,(q4)\,s\,(q6) - c\,(q4)\,c\,(q5)\,c\,(q6) & c\,(q6)\,s\,(q4) + c\,(q4)\,c\,(q5)\,s\,(q6) & -c\,(q4)\,s\,(q5) \\ -c\,(q4)\,s\,(q6) - c\,(q5)\,c\,(q6)\,s\,(q4) & c\,(q5)\,s\,(q4)\,s\,(q6) - c\,(q4)\,c\,(q6) & -s\,(q4)\,s\,(q5) \\ -c\,(q6)\,s\,(q5) & s\,(q5)\,s\,(q6) & c\,(q5) \end{bmatrix} \tag{2.24}$$

where:

$$s = sin \qquad and \qquad c = cos$$

Evaluating matrix 2.24, expressions for $q4$, $q5$ and $q6$ are found:

$$q5 = cos^{-1}(\boldsymbol{R}_6^3(3,3)) \tag{2.25}$$

$$q4 = tan^{-1}\left(\frac{-\boldsymbol{R}_6^3(2,3)}{-\boldsymbol{R}_6^3(1,3)}\right) \tag{2.26}$$

$$q6 = tan^{-1}\left(\frac{-\boldsymbol{R}_6^3(3,2)}{-\boldsymbol{R}_6^3(3,1)}\right) \tag{2.27}$$

When implementing the inverse kinematics in MATLAB®, the ATAN2 function is used instead of tan^{-1} . This assures that the solution is in the correct quadrant. The MATLAB®script used to calculate the inverse kinematic is found in the appendix A.3.

2.1.3 Jacobian Matrix

While the forward kinematics are functions that describe the relationship between the position and orientation of the end-effector and the joint values, the Robot Jacobian of this function, the Jacobian matrix, describes the velocity relationship. The Jacobian is important when modeling or controlling robots in path planning, looking for singularity configurations and to transform torques and forces applied to the end-effector into joint torques [2].

For a robot with n joints, the joint variables $q_1, ..., q_n$ and

$$\boldsymbol{T}_n^0(q) = \begin{bmatrix} \boldsymbol{R}_n^0(q) & \boldsymbol{o}_n^0(q) \\ 0 & 1 \end{bmatrix} \tag{2.28}$$

The Jacobian has n columns which are found as:

$$\boldsymbol{J}_i = \begin{bmatrix} \boldsymbol{J}_{v_i} \\ \boldsymbol{J}_{w_i} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{i-1}^0 \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (\boldsymbol{o}_n^0 - \boldsymbol{o}_{i-1}^0) \\ \boldsymbol{R}_{i-1}^0 \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} \tag{2.29}$$

where:

$\boldsymbol{J}_{v_i}$	-	Describes the Translational Velocity Relationship.
$\boldsymbol{J}_{w_i}$	-	Describes the Rotational Velocity Relationship.

Then the Jacobian will be:

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_1 & . & . & . & \boldsymbol{J}_n \end{bmatrix} \tag{2.30}$$

For the 6-DOF robot the Jacobian will take this form:

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_v \\ \boldsymbol{J}_w \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_0^0 \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (\boldsymbol{o}_6^0 - \boldsymbol{o}_0^0) & \cdots & \boldsymbol{R}_5^0 \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (\boldsymbol{o}_6^0 - \boldsymbol{o}_5^0) \\ \boldsymbol{R}_0^0 \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \cdots & \boldsymbol{R}_5^0 \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} \tag{2.31}$$

The MATLAB®script used to calculate the Jacobian matrix is found in the appendix A.6.

2.2 Robot Stiffness

6-DOF industrial robots are, due to their versatility, commonly used for a large variety of tasks and processes. Some examples are welding and pick and place applications. Compared to traditional CNC machines, industrial robots are prone to higher deflections when exposed to TCP load forces. This is the main reason why industrial robots are not yet commonly used in precision contact applications.

In some precision contact applications, versatility are of high importance, hence industrial robots are considered. This applies to the milling application described in this report. Although high stiffness is most critical when dealing with hard materials, it also has to be taken into account when dealing with soft materials where the rate of material removal is high [6]. In order to improve the kinematic and elastostatic performance, the stiffness has to be estimated and compensated for. The combined joint stiffness is a combination of link stiffness, bearings, motors, gears and controller gains [6] [7].

Different approaches of estimating the robot stiffness have been examined over the years. In [7] it is assumed that the first three joints have the largest impact on the end effector position, and the stiffness is calculated for these joints based on given data for motors and gears. Another possible method described in [8] is to clamp all the joints except for the one evaluated, and evaluate the stiffness individually. The process is repeated for all the joints.

In [9] a method applicable on all 6-DOF industrial robots, based on the conservative congruence transformation is described. Different robot configurations are chosen in order to have good dexterity, and to avoid configurations where the complementary stiffness matrix $\boldsymbol{K}_c$ have too much influence on the Cartesian stiffness matrix $\boldsymbol{K}_x$. This method is considered a global approach, meaning that the forces are applied to the end effector. During this experiment, both forces and end effector poses are logged, and the stiffness is calculated based on these measurements. This method do not require any change in the experimental set-up, it does however include possible link deformations and will depend on robot configuration. In addition, the stiffness for several joints is evaluated in one operation [6].

A combined local/global approach is described in [6]. A local method is used to identify the stiffness for the joints 1, 2 and 6. This is achieved by measuring the displacements locally with a laser tracker and reflector. The load is applied to the end-effector, and is measured with a

6-DOF force/torque sensor mounted between the end-effector and the robot mounting flange. The remaining three joints are evaluated by the means of a combined local/global method. This is achieved by using calculation on the measured end-effector deflection in combination with the already determined joint values.

The stiffness analysis conducted in this project is based on the method described in [6]. However, the methods used are altered, in an attempt to both simplify the procedure and improve the results. The influence of pre-loaded joints in the stiffness analysis is also considered, and to avoid backlash uncertainties, the joints in question were initially pre-loaded.

The results from this analysis were then used together with the method described in [9], to find configurations where $\boldsymbol{K}_c$ has least influence on $\boldsymbol{K}_x$, and therefore can be ignored. Configurations where the robot has good dexterity, was investigated by evaluating the condition number of the normalized Jacobian matrix $\boldsymbol{J}_n$, like explained in [9]. Based on these analyses and evaluations an appropriate region of operation for the robot was defined, and the robot position relative to the work object could be determined.

The Jacobian matrix $\boldsymbol{J}$ can be used to calculate the change in end effector pose $\delta\boldsymbol{d}$ related to rotation in the robot joints $\delta\boldsymbol{q}$:

$$\delta\boldsymbol{d} = \boldsymbol{J} \cdot \delta\boldsymbol{q} \tag{2.32}$$

The Jacobian matrix $\boldsymbol{J}$ can also be expressed as:

$$\boldsymbol{J}_{i,j} = \frac{\delta\boldsymbol{d}_1}{\delta\boldsymbol{q}_j}, \qquad (i = 1,..,6 \qquad j = 1,..,6) \tag{2.33}$$

The joint torques $\boldsymbol{\tau}$ related to the wrench, $\boldsymbol{W}$, which contains external forces and moments, can also be calculated with the help of $\boldsymbol{J}$:

$$\boldsymbol{\tau} = \boldsymbol{J}^T \cdot \boldsymbol{W} \tag{2.34}$$

The relation between $\delta\boldsymbol{d}$, $\boldsymbol{W}$ and the robot Cartesian stiffness matrix $\boldsymbol{K}_x$ can be expressed as:

$$\boldsymbol{W} = \boldsymbol{K}_x \cdot \delta\boldsymbol{d} \tag{2.35}$$

And the relation between $\boldsymbol{q}$, $\boldsymbol{\tau}$ and the joint stiffness matrix $\boldsymbol{K}_\theta$ is expressed as:

$$\boldsymbol{\tau} = \boldsymbol{K}_\theta \cdot \delta\boldsymbol{q} \tag{2.36}$$

Partial differentiation of equation 2.34 with respect to $\boldsymbol{q}$ gives:

$$\frac{\delta\boldsymbol{\tau}}{\delta q} = \frac{\delta\boldsymbol{J}^T}{\delta q} \cdot \boldsymbol{W} + \boldsymbol{J}^T \cdot \frac{\delta\boldsymbol{W}}{\delta\boldsymbol{d}} \cdot \frac{\delta\boldsymbol{d}}{\delta q} \tag{2.37}$$

In [10], the complementary stiffness matrix $\boldsymbol{K}_c$ is defined as:

$$\boldsymbol{K}_c = \frac{\delta\boldsymbol{J}^T}{\delta q} \cdot \boldsymbol{W} \tag{2.38}$$

Substituting equations 2.34, 2.35 and 2.38 into equation 2.37 gives:

$$\boldsymbol{K}_x = \boldsymbol{J}^{-T} \cdot (\boldsymbol{K}_\theta - \boldsymbol{K}_c) \cdot \boldsymbol{J}^{-1} \tag{2.39}$$

If it is assumed that the influence of $\boldsymbol{K}_c$ so small that it can be neglected, equation 2.39 is reduced to:

$$\boldsymbol{K}_x \approx \boldsymbol{J}^{-T} \cdot \boldsymbol{K}_\theta \cdot \boldsymbol{J}^{-1} \tag{2.40}$$

The stiffness in each joint (q) is simplified and illustrated in figure 2.7. K_θ is the joint stiffness $[\frac{Nm}{rad}]$, F is the load acting on the robot arm, and θ is the angular displacement caused by the load. L_1 is the length from the joint origin to where the force is acting.

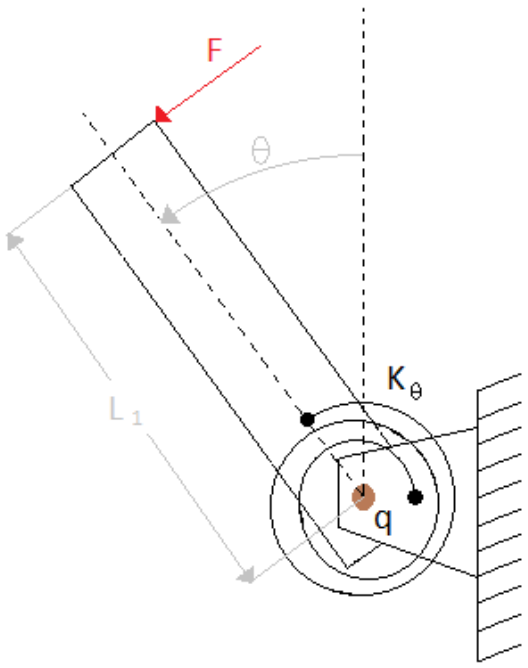


Figure 2.7: Torsional Spring

The equation for torsional stiffness:

$$K_\theta = \frac{F \cdot L_1}{\theta} \tag{2.41}$$

In order to calculate the stiffness, the angular displacement has to be calculated:

$$sin(\theta) = \frac{L_d}{L_2} \tag{2.42}$$

When calculating the angular deflections in a robotic stiffness analysis, the angles are small. Therefore the small-angle approximation can be used:

$$sin(\theta) \approx \theta \tag{2.43}$$

Hence, equation 2.42 can be simplified to:

$$\theta = \frac{L_d}{L_2} \tag{2.44}$$

Combining equation 2.41 and equation 2.44 we get:

$$K_\theta = \frac{F \cdot L_1 \cdot L_2}{L_d} \tag{2.45}$$

where:

F	–	Applied Load Force $[N]$
L_1	–	Moment Arm of the Applied Load Force $[m]$
L_2	–	Distance from Joint Origin to where Displacement is Measured$[m]$
L_d	–	Measured Displacement$[m]$

2.3 Sensors

Throughout this section the different techniques which can be used to locate the door will be discussed. Both the Zivic camera and the lidar solution has not been further investigated due to hardware and software restrictions. The probing, photelectric sensor, and camera solution is presented, where these are feasible solutions to our sensor requirements.

Probing

A probing approach is often used to locate objects with high accuracy. This process often needs initial information or installation to correctly locate the object. It is usually used to determine work-piece length in CNC machines or locate milling objects for industrial robots. The detection of the given object is often achieved by using a switch or an analogue sensor to detect the work-piece. Thereafter it returns a signal to the controller verifying the located obstacle's length.

Lidar

Lidars could also be a feasible solution to the problem. The pose of the door can be calculated by using one or two lidars positioned at a known angle to the door. This solution would be quick and reliable. However, a lidar uses time of flight of light. The sensor can measure

the time between emitted and received signal. By knowing the light velocity and the time it takes to return to sensor, the distance from the sensor to the object can be calculated. With the light velocity being approximately 3^8 m/s, the lidar struggle to distinguish between small distance changes, seeing the time between the return of these two signals are extremely small. Consequently, the lidar accuracy is not good enough to reach the desired accuracy of $\pm 0.5mm$. Thus, the lidar does not meet the projects criterion.

Photoelectric Sensor

To achieve high accuracy in the area of micrometres, photoelectric sensors can be used, such as the OPT, shown in appendix A.20. The sensor use an optical triangulation method to determine a point on an object. The sensor focuses the returning light onto a 2-segment photodiode. The 2-segment diode consists of a near (N) and far (F) segment. If both the segments return the same signal, the object is at its Set-Distance. The distance the received light is focused away from the Set-Distance (labelled a) is proportional with the displacement distance of the obstacle (labelled A) [4]. The sensors with its internal components are shown in figure 2.8.

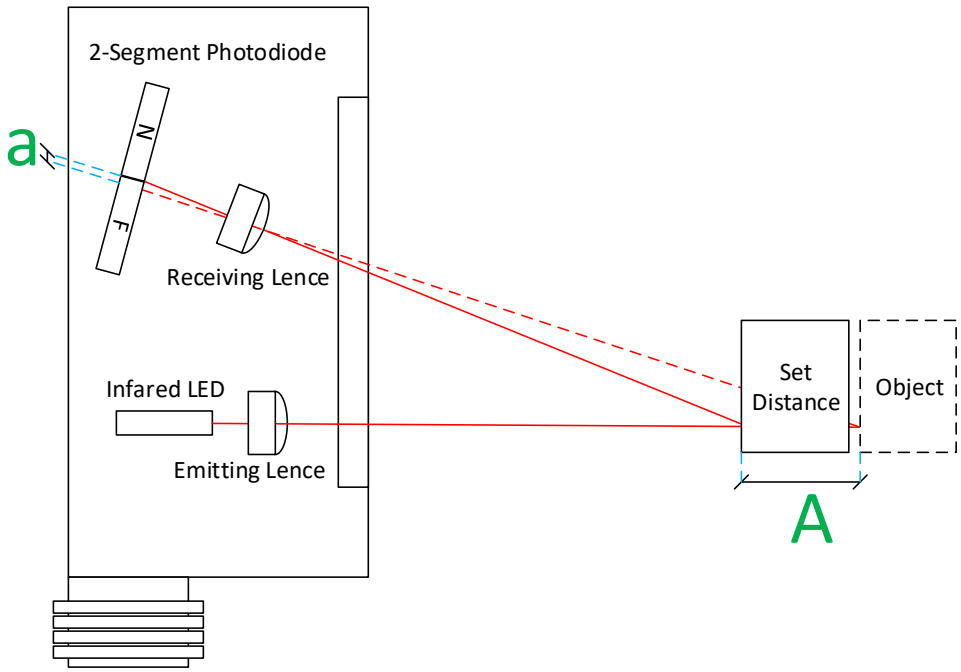


Figure 2.8: Photoelectric Sensor

Camera

The camera used throughout this report is a pin-hole camera; it functions by using individual pixels, which detect the RGB value of the reflected light. The number of these pixels results in the resolution of the camera. The principle of the pin-hole camera is shown in figure 2.9.

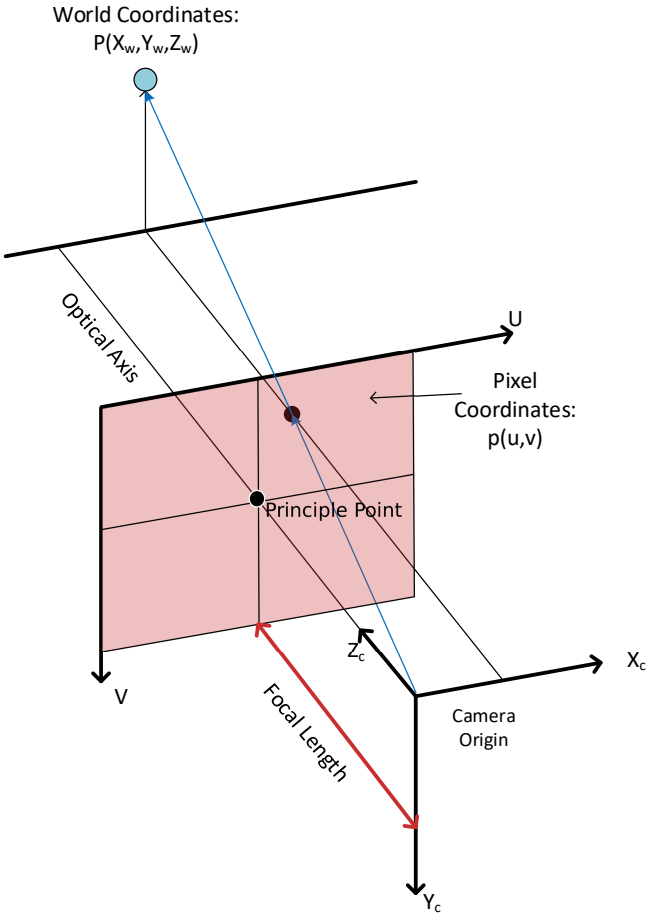


Figure 2.9: Pin-hole Camera

When the data is retrieved from the camera, different algorithms can be used to determine features such as lines and different geometries in the picture. Multiple cameras can also be used to extract depth information in a picture.

Camera Calibration

When using cameras for industrial purposes where accuracy requirements are high, it is often necessary to calibrate the camera to identify both the intrinsic parameters and lens distortion factors. For this project, the MATLAB®camera calibration toolbox is used, together with a planar grid, like a chequerboard, where the square dimensions are known. This method requires several pictures of the grid from different positions and orientations. The parameters are determined by solving a least square linear minimization followed by a non-linear refinement [5]. The intrinsic parameters are given in a $3x3$ matrix, referred to as the $\boldsymbol{k}$ matrix, and holds information about the internal parameters of the camera (focal length in two directions and coordinates of the principal point) shown in figure 2.9. The distortion factors are given as two or three coefficients which can be used to compensate for lens distortion. Lens distortion is commonly seen as barrel distortion or pincushion distortion illustrated in figure 2.10.

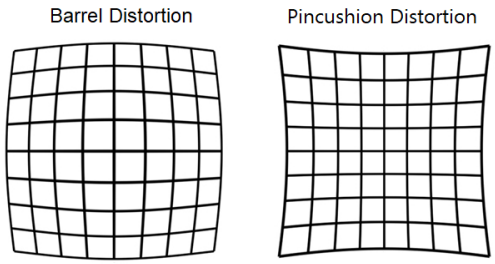


Figure 2.10: Lens Distortions

Zivid camera

The industry is always pushing the envelope in terms of speed and accuracy. One of the leading camera technologies is utilized in the Zivid camera. In a range from 0.6 to 1m, the Zivid camera can produce a point cloud with 0.1mm accuracy, with an update frequency of 10Hz, according to the data sheet which can be found in the appendix A.22. The camera works in such a way that it simultaneously emits a grid with a projector and takes pictures. This technique allows for detection of deformations in the grid, which again can be calculated to position. The entire operation of locating the pose of the door can done with only one camera placed from the door as seen in figure.2.11. The Zivid camera price is not available. However, this camera is a good solution in terms of installment cost, accuracy and time.

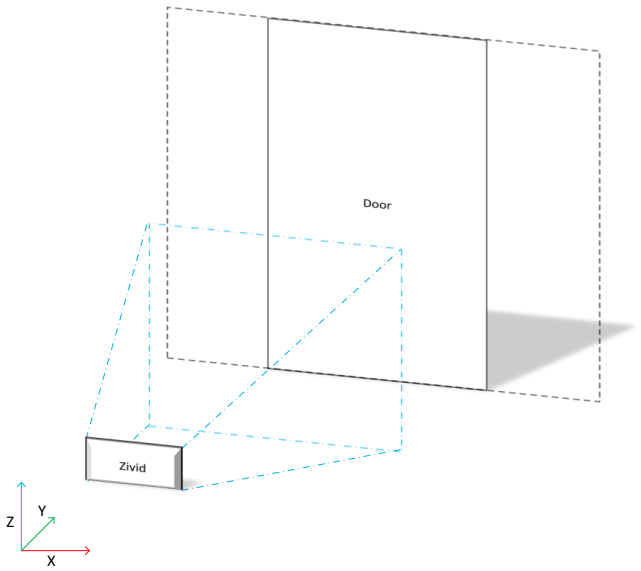


Figure 2.11: Zivid Camera Configuration

2.4 Path Correction

When using robots in contact applications, such as milling, the deflection caused by the forces exiting the TCP of the robot can no longer be neglected [11]. The deflection is related to the external force and torque applied to the end-effector, the configuration of the robot (inertia and moment) and the overall robot stiffness. The stiffness of an industrial robot can be calculated as shown in section 3.1. When the stiffness of the robot is determined, the deflections can be compensated for. This is achieved by adding the calculated deflection to the TCP in the different target points of the milling operation. There are several approaches to correcting the robot path trajectory, such as:

Closed-loop Sensor Correction

By using a sensor to track the TCP, the global pose of the spindle can be calculated. By using this data, a closed loop controller can be designed to compensate for the deflections caused by the external forces [11]. This technique is highly effective when there is a large variety in milling paths and milling material. It introduces the need of expensive 6-DOF tracing equipment and a well tuned controller. Robot milling applications are prone to oscillations which causes noise in laser measurements. To reduce the effect of the noise, filtering is necessary e.g. low-pass filter or a Kalman filter. Further, the mean value of the signal must be used to achieve the best possible results. However, neither a force-sensor nor the stiffness analysis is needed using this method.

On-line Correction

On-line correction requires both a force-sensor and stiffness data. Like the sensor compensation it compensates real-time, by using force-sensor data together with stiffness and configuration data to compensate for the tool displacement. Like the sensor compensation, this solution is desirable if the need for a large variety of paths and milling material is used. However, it requires a tuned closed-loop controller to function, and this tuning can be expensive [11].

Off-line Correction

Off-line correction can be used to avoid any external real-time feed-back system. This method is often used when both the milling material and paths are repeated in the production process, or when implementation of a closed-loop controller is not possible. Using off-line correction, the process forces must be identified prior to the milling sequence. This is necessary seeing this data is used to correct the actual path. Off-line correction applies to the desired approach discussed throughout this thesis.

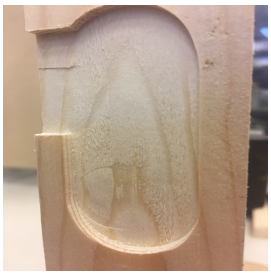
2.5 Milling

There are many parameters which influence the feeds and speeds of the mill. These parameters are dependent on many different factors,e.g. work-piece clamping, mill stick-out, work-piece material, mill flutes, machine rigidity, climb/conventional milling [6]. Throughout this section, a closer insight to these factors will be presented, and the effect they have on the final feed and speed determination [12].

Work-piece Clamping

Without sufficient clamping of the work-piece; the forces exerted to the material must be reduced. Thus, there exists a large variety of clamping methods to secure the work-piece to validate it as rigid. For roughing passes, the material removal rate is often high, therefore rigid clamping is of high importance. The results of excessive feeds on weak clamping can cause quite unsatisfactory results. In figure.2.12a. and figure.2.12b. the same feed on well and weak clamping is shown, respectably.

(a) Weakly Clamped Work-Piece



(b) Sufficiently Clamped Work-Piece

Figure 2.12: Comparing Weak and Sufficient Work-Piece Clamping

Tool Stick-out

Tool stick-out is the length from the spindle to the working area of the milling tool. A large stick-out often requires reduced feeds to reduce the load of the tool. If the load on the milling tool is too high, it will deflect, and cause fluctuations, resulting in excessive tool wear. Throughout this project, the milling tool stick-out is kept to a minimal seeing there will be no cutting of deep features. However, if deep features are desirable, the forces exerted on the tool must be reduced to avoid fluctuations.

Work-piece Material

If the work-piece material is considered a hard material, the demand for a rigid machine is of high importance if high removal rate is desirable. In the industry, (after calculating conservative starting values) these limits are often determined by utilizing empirical tests. This is achieved by increasing the feed and speed of the machine until unsatisfactory surface finish is observed. The desired feed and speed is then determined by reducing the feed and speed slightly to achieve

a high removal-rate as well as desired surface finish. The work-piece material also determines the relation between feed-rate and spindle speed and the tool choice.

Number of Flutes

The number of flutes on the chosen mill has a direct relation with the feed-rate. All tooling has a desirable chip load (CL)[1] which is important to fulfil to not damage the mill or work harden [2]the Work-piece.

Machine Rigidity

If both the work-piece is secured sufficiently and the stick-out is kept to a minimum, the rigidity of the machine is the next interest-point. If the machine is very stiff, large martial-removal rates can be achieved,i.e. A conventional mill is estimated to be rigid, whereas a router is not. If the forces exerted to the mill exceed the machines capabilities, the end-product will suffer from uneven surface finish or tool breakage may occur. These are results of mill fluctuations or work-hardening.

Climb/Conventional Milling

Climb milling is used to achieve an even surface finish, transfer generated heat into chips, and allow for good chip evacuation. This method is often used on modern machines, where anti backlash lead-nuts are installed, or other anti-backlash techniques are utilized. Conventional milling has its origin from milling with manual machines. The largest advantage using conventional milling is the fact that the technique pushes the mill in the opposite direction of the feed direction. This results in a constant engagement of the lead-screw, which removes the influence of back-lash. However, the method pushes chips in front of the tool, resulting in an uneven finch. Also, the heat produced is transferred into the work-piece. The two methods can be seen in figure 2.13.

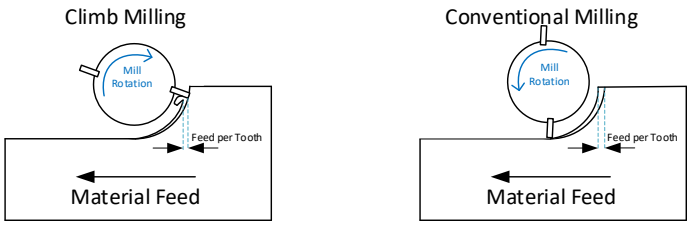


Figure 2.13: Conventional and Climb Milling

[1] *IPT - Also called Inch Per Tooth, chip load per tooth (CLPT), or simply chip load is the amount of material being removed by each flute of the cutting tool per revolution.*

[2] *Work hardening of materials is a condition that should be avoided while machining. It is caused when heat generated by the cutting tool transfers to the work piece material and causes plastic deformation. The process is similar to a heat treatment of the workpiece but on a lower scale.* [13]

Regardless of method; it is important to evaluate the chips generated by the milling sequence,e.g. Wood milling should produce large chips, without any burn marks. If either dust or burn marks are produced, this is a clear indication of miscalculated feeds and spindle-speeds.

2.6 Safety

When working with industrial robots, safety is important to consider. An industrial robot is often programmed to follow a specific path. Consequently, the robot will strive to follow this path, despite obstacles. A valid alternative to robots which one can interact with, is collaborative robots[3] In this application, where the robot is used for a milling application, the need of additional safety is often required. One should not be, nor allowed to be within the working space of the robot while it is active.

A possible solutions are fences to isolate the robot. Fences contain locks with internal switches, if the switch is triggered, the motion stops immediately. Such a solution is vastly used in industry concerning robots applications, and can be seen in figure.2.14.

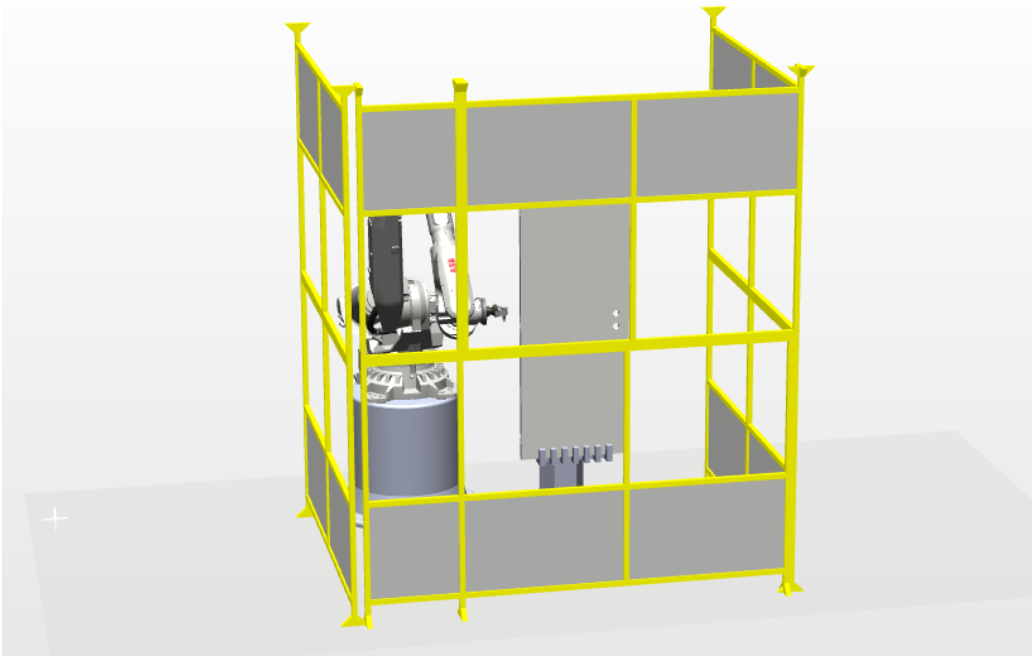


Figure 2.14: Robot Cell with Fence

Another solution, often used in pick-and-place applications, is laser curtains. This solution requires less installments, and like the fence, it stops the robot immediately if the robot region is entered. However, it does not stop any loose objects which may come from the robot cell. This solution can be seen in figure.2.15.

[3] A collaborative robot, is a robot that senses if unexpected load is applied to the joints, and will stop if this is the case. It often moves slower than a normal industrial robot, and it is usually more flexible and less stiff. A collaborative robot is designed to work in unison with humans.[http://new.abb.com/products/robotics/industrial-robots/yumi]

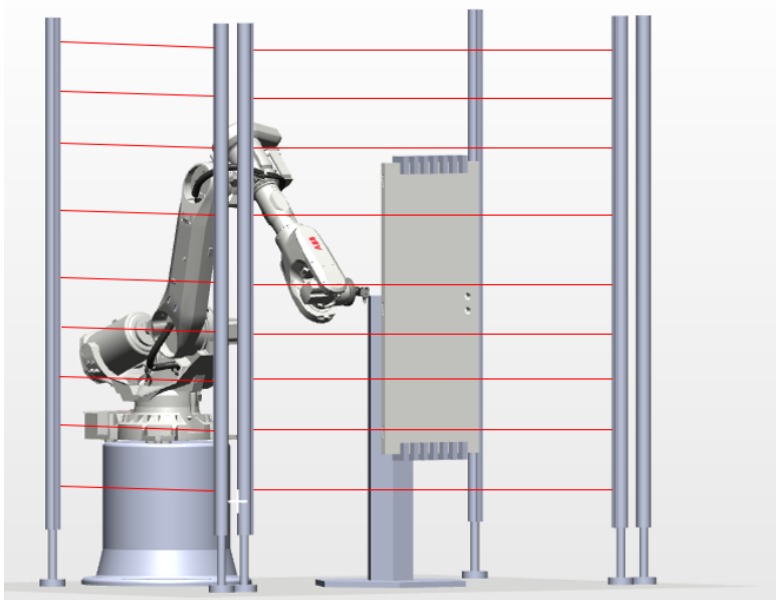


Figure 2.15: Robot Cell with Lasers

A more modern approach to robot safety is camera surveillance. Here, the cameras track the robot, and algorithms are used to distinguish between robot, static obstacles and unwanted obstacles e.g. humans. This solution can allow for a smaller cell, as well as an inexpensive instalment cost. It can also allow for an industrial robot to become somewhat collaborative.

In the robot cell used for this project, Plexiglass is installed to separate the operator from the robot. This solution allows the operator good visibility as well as high safety. It also stops chips and dust from leaving the respective cell while milling. The installation can be seen in figure 2.16.

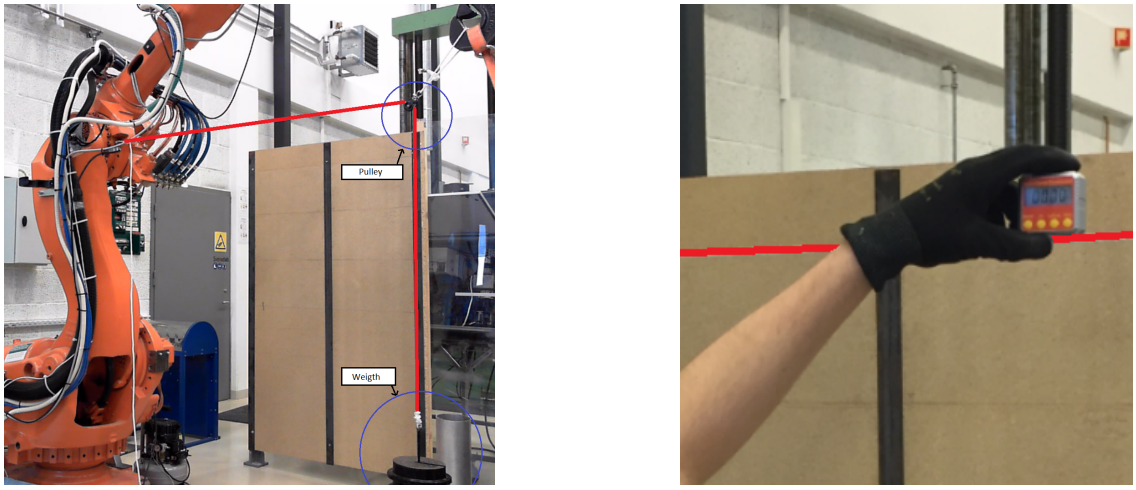
Figure 2.16: Robot Cell with Plexiglass

3 | Method

The different methods proposed and used in this thesis are presented in this chapter. First, a stiffness identification method is presented, followed by an evaluation of the influence of pre loading the joints in the stiffness analysis. The influence of link stiffness compared to joint stiffness is also included. A simulation model of the robot in SimulationX, used to evaluate the results of the stiffness analysis is also presented. Selecting region of operation based on different door sizes, robot kinematics and kinematic performance is described, followed by description of methods for localization of work-object with several different sensors techniques. This chapter also describes how door specifications listed in a database can be used to generate a robot path trajectory and desired milling action.

3.1 Stiffness Identification

As mentioned in section 2.2, the stiffness analysis performed for this project is based on the method used in [6]. However, some changes are made in an attempt to simplify the analysis and improve the results. For this project, joint configurations are chosen to acquire a high degree of isolation for the evaluated joint. Another way to isolate a joint is to use several reflector positions, and do measurements on both sides of a joint with the robot both loaded and unloaded. This approach is used in the evaluation of joint 4 and 5. In addition, the force used to evaluate joint 2 is attached directly in joint 3, in order to eliminate any contribution from joint 3. In order to reduce the measurement noise, the forces are applied using a rope, pulley and weights, like shown in figure 3.1a, instead of being manually applied. Figure 3.1b shows how the direction of the force was adjusted with a digital angel-sensor.



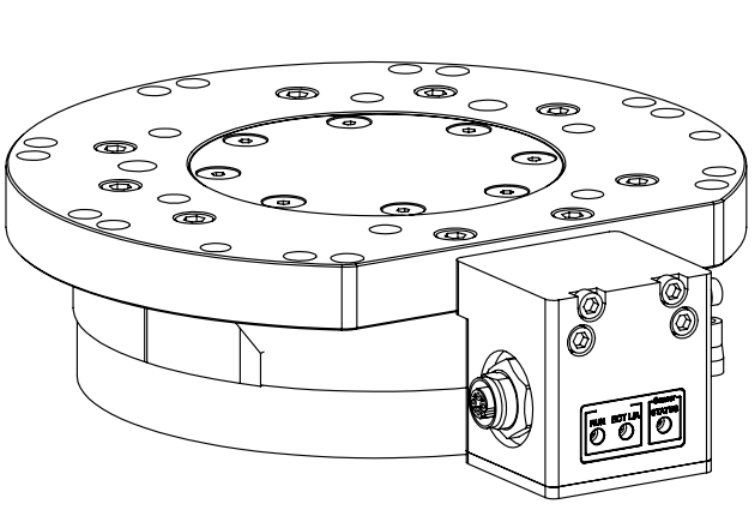
(a) Applying Force using Line, Pulley and Weights (b) Adjusting the Load Force Direction

Figure 3.1: Applying Load for Evaluation of Joint 2

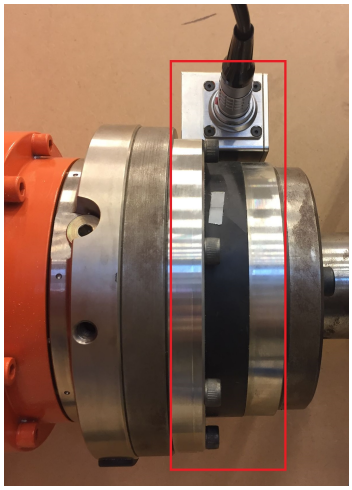
The proposed approach combines both local and global methods, and is conducted with the robot in home position, as defined in section 2.1. The exceptions are for joint 2, where $q_3 = -\frac{\pi}{6}$ rad, joint 3, where $q_5 = \frac{\pi}{2}$ rad, for joint 4, where $q_4 = \frac{\pi}{2}$, $q_5 = -\frac{\pi}{2}$ and $q_6 = \frac{\pi}{2}$, and finally for joint 6, where $q_5 = \frac{\pi}{2}$.In order to include the controller gains in the analysis, the joint brakes were released, and the joints pre-loaded in all the experiments.

In order to see the impact of pre-loading the joints in the joint stiffness analyses, and then be able to determine in which configurations and joint movements the results are valid, an additional experiment was conducted on joint 1.

In this project, an ABB IRB 6600/175-2.55 is used, but the method proposed is applicable to other 6-DOF robots with similar kinematic design. For all experiments, except the evaluation of joint 2 where the force is applied directly to joint 3, the external forces were applied to the end effector. The wrench was measured by a flange mounted ATI Omega 160 IP60 6-DOF force/torque sensor showed in figure 3.2a, and in the red square in figure 3.2b. The ATI Omega 160 has a resolution of 0.25-0.5 N and 0.0125-0.05 Nm. The load used for all the experiments is a weight of 20.120 kg



(a) ATI Omega160 IP60 Sketch

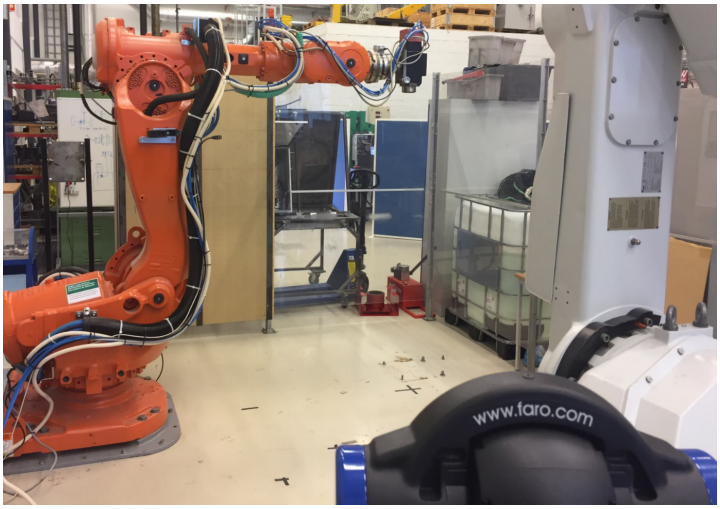


(b) ATI Omega160 IP60 F/T Mounted on Robot Flange

Figure 3.2: The Force/Torque Sensor used for the Experiments

The displacements were measured by a FARO Xi laser tracker and reflector shown in figure 3.3a. The white arrow points at the reflector in figure 3.3a, where it is placed to initialize the Faro. This laser tracker/reflector set-up has a resolution of 10-30 μm. Figure 3.3b shows the robot from the position of the Faro. Both the Faro and ATI were operated on the same computer and measurements were stored as *$.csv$ and *$.txt$ files. The results were analysed using MATLAB®, and SimulationX was used to simulate and validate the results in the experiments.

(a) The Faro Laser Tracker

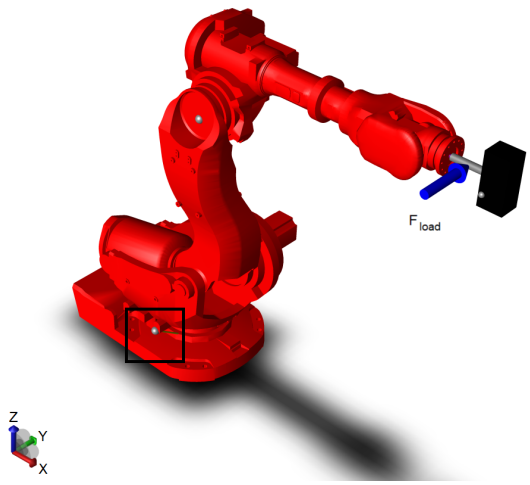


(b) Robot View from Faro Laser Tracker

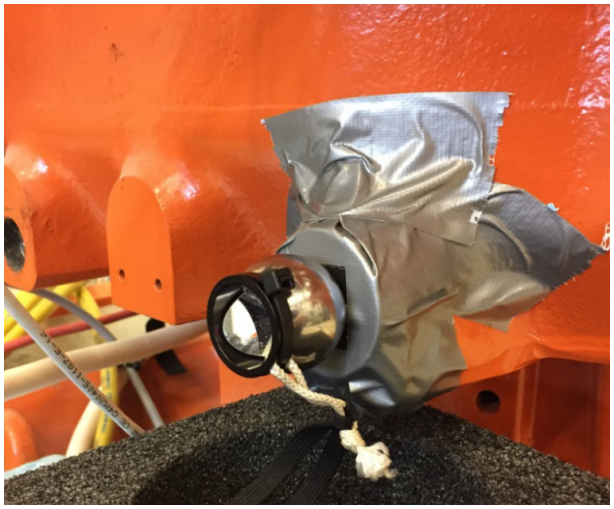
Figure 3.3: Stiffness Experiments Set Up

3.1.1 Joint 1

The stiffness of joint 1 is identified by measuring the local joint displacement caused by a load, while the robot is in home position. A reflector is placed on the robot base as shown in the black square in figure 3.4a, and 3.4b. The reflector position is logged using the laser tracker, while applying a force to the end effector, as shown with the blue arrow in figure 3.4a. The applied load is logged with the ATI Omega 160.



(a) Applied Force Identifying Stiffness Joint 1



(b) Reflector Placement on Robot Base

Figure 3.4: Stiffness Identification Joint 1

The joint stiffness in joint 1 is then found:

$$K_1 = \frac{F_{load} \cdot L_1 \cdot L_2}{L} \tag{3.1}$$

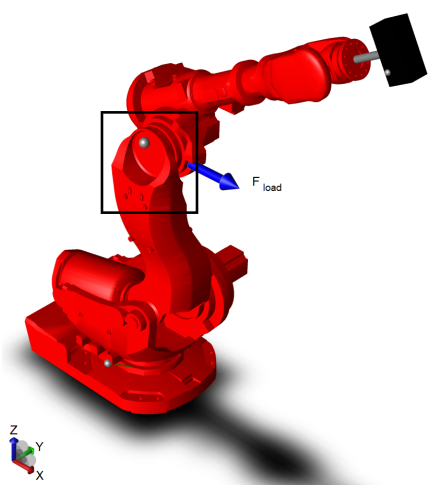
$$L = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} \tag{3.2}$$

where:

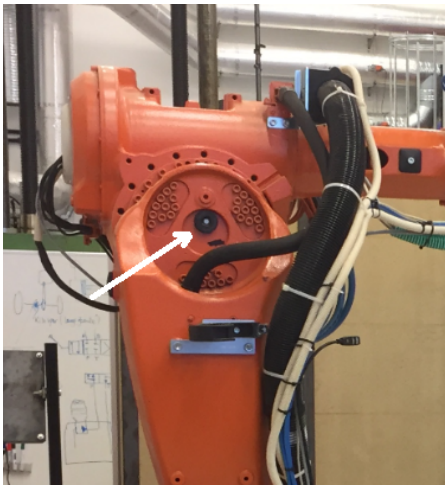
$$\begin{aligned} L_1 &= \text{Moment arm of the load applied} \\ L_2 &= \text{Distance from joint 1 to reflector position (radius)} \\ x, y, z &= \text{Reflector position measured by laser tracker} \\ x_0, y_0, z_0 &= \text{Reflector position measured by laser tracker (unloaded robot)} \end{aligned}$$

3.1.2 Joint 2

The stiffness of joint 2 is also identified by measuring the local joint displacement caused by a load. The robot is in home position except from joint 3 which is jogged to $-\frac{\pi}{6}$ rad. A reflector is placed on the robot in the "joint 3 ring" as shown in the black square in figure 3.5a, and in figure 3.5b. The reflector position is logged using the laser tracker. A force is applied directly in the joint 3 connection as shown with the blue arrow in figure 3.5a. This load is not connected through the ATI Omega, but directly to the robot using a rope and a pulley.



(a) Applied Force Identifying Stiffness Joint 2



(b) Reflector Placement on Robot

Figure 3.5: Stiffness Identification Joint 2

The joint stiffness in joint 2 is then found:

$$K_2 = \frac{F_{load} \cdot L_1 \cdot L_2}{L} \tag{3.3}$$

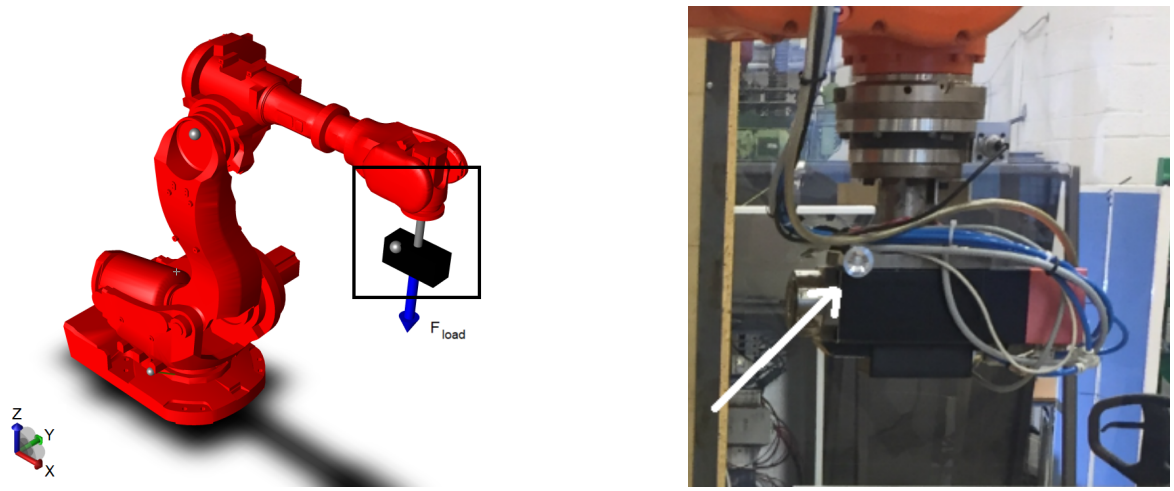
$$L = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} \tag{3.4}$$

where:

L_1	–	Moment Arm of the Applied Load Force
L_2	–	Distance from Joint 2 to Reflector Position (Radius)
x, y, z	–	Reflector Position Measured by Laser Tracker
x_0, y_0, z_0	–	Reflector Position Measured by Laser Tracker (Unloaded Robot)

3.1.3 Joint 3

The approach for identifying the stiffness in joint 3 is a combination of local and global methods. The robot is in home position except from joint 5 which is jogged to $\frac{\pi}{2}$ rad. A reflector is placed on the robot end effector as shown in the black square in figure 3.6a, and in figure 3.6b. The reflector position is logged using the laser tracker. A force is applied to the end-effector as shown with the blue arrow in figure 3.6a. The applied load is logged with the ATI Omega 160. In addition to cause a angular deflection in joint 3, the applied load will also cause a small angular deflection in joint 2. Since the stiffness for joint 2 is already analysed, the measured end effector deflection caused by joint 2 can be calculated and subtracted from the laser tracker measurements.



(a) Applied Force Identifying Stiffness Joint 3 (b) Reflector Placement on Robot

Figure 3.6: Stiffness Identification Joint 3

The joint stiffness in joint 3 is then found:

$$K_3 = \frac{F_{load} \cdot L_1 \cdot L_2}{L} \tag{3.5}$$

$$L = \sqrt{(x - x_0 - d_{2,x})^2 + (y - y_0 - d_{2,y})^2 + (z - z_0 - d_{2,z})^2} \tag{3.6}$$

$$\boldsymbol{d_2} = \boldsymbol{J} * [0 \quad \frac{\tau_2}{K_2} \quad 0 \quad 0 \quad 0 \quad 0]^T \tag{3.7}$$

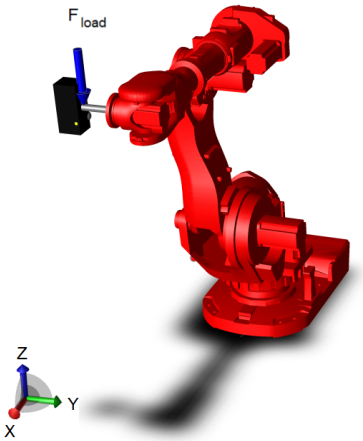
$$\boldsymbol{\tau} = \boldsymbol{J} * \boldsymbol{W} \tag{3.8}$$

where:

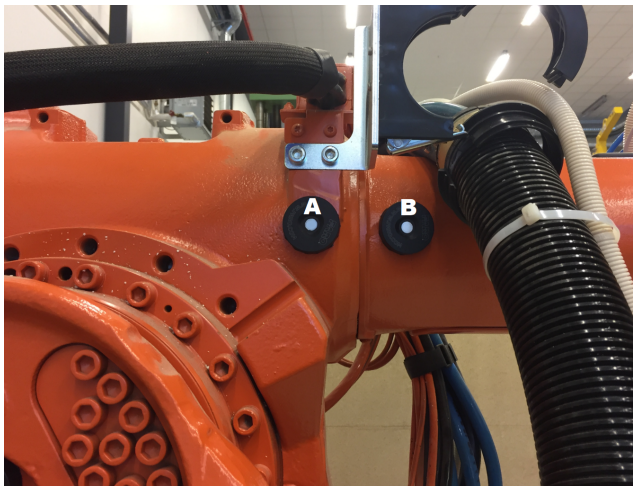
L_1	–	Moment Arm of the Load Applied
L_2	–	Distance from Joint 3 to Reflector Position (Radius)
x, y, z	–	Reflector Position Measured by Laser Tracker
x_0, y_0, z_0	–	Reflector Position Measured by Laser Tracker (unloaded robot)
$\boldsymbol{d}_2$	–	Displacement Caused by Joint 2
$\boldsymbol{\tau}$	–	Torque Vector
$\boldsymbol{J}$	–	Jacobi Matrix
$\boldsymbol{W}$	–	Wrench from ATI Omega 160

3.1.4 Joint 4

To identify the stiffness in joint 4, the approach is to measure the local angular deflection in the joint. The robot is in home position except from joint 4 which is jogged to $-\frac{\pi}{2}$ rad, joint 5 is jogged to $\frac{\pi}{2}$ rad, and joint 6 is jogged to $\frac{\pi}{2}$ rad. Two reflector pucks are placed on each side of the joint, A and B, as shown in figure 3.7b. Positions are logged with the reflector attached to both pucks, with the robot both unloaded and loaded, using the laser tracker. A force is applied to the end effector as shown with the blue arrow in figure 3.7a. The applied force is logged with the ATI Omega 160. In addition to cause a angular deflection in joint 4, the applied load will also cause a small angular deflection in joint 2 and 3. The contribution from joint 2 and 3, is therefor accounted for in the estimation of joint 4.



(a) Applied Force Identifying Stiffness Joint 4



(b) Reflector Placement on Robot while Identifying Stiffness Joint 4

Figure 3.7: Stiffness Identification Joint 4

To estimate the stiffness in joint 4, first the vector from A to B both for loaded and unloaded robot are found:

$$\boldsymbol{AB}_u = [(B_{x_0} - A_{x_0}) \quad (B_{y_0} - A_{y_0}) \quad (B_{z_0} - A_{z_0})] \tag{3.9}$$

$$\boldsymbol{AB}_l = [(B_x - A_x) \quad (B_y - A_y) \quad (B_z - A_z)] \tag{3.10}$$

$$|\boldsymbol{AB}_u| = \sqrt{(\boldsymbol{AB}_{u,x})^2 + (\boldsymbol{AB}_{u,y})^2 + (\boldsymbol{AB}_{u,z})^2} \tag{3.11}$$

$$|\boldsymbol{AB}_l| = \sqrt{(\boldsymbol{AB}_{l,x})^2 + (\boldsymbol{AB}_{l,y})^2 + (\boldsymbol{AB}_{l,z})^2} \tag{3.12}$$

Then the angular difference between $\boldsymbol{AB}_u$ and $\boldsymbol{AB}_l$ can be calculated:

$$\phi = cos^{-1}\left(\frac{\boldsymbol{AB}_u \cdot \boldsymbol{AB}_l}{|\boldsymbol{AB}_u| \cdot |\boldsymbol{AB}_l|}\right) \tag{3.13}$$

Next, the contribution from joint 2 and 3 to the angular difference is calculated:

$$\alpha = \left| tan^{-1}\left(\frac{\sqrt{(A_x - A_{x_0})^2 + (A_y - A_{y_0})^2 + (A_z - A_{z_0})^2}}{L_2}\right)\right| \tag{3.14}$$

Using geometry the curvature c of joint 4, is found:

$$c = |\boldsymbol{AB}_u| \cdot tan(\phi - \alpha) \tag{3.15}$$

And then the angular displacement θ in joint 4, is found:

$$\theta = tan^{-1}\left(\frac{c}{R_4}\right) \tag{3.16}$$

Finally the stiffness, K_4 can be found:

$$K_4 = \frac{F_{load} \cdot L_1}{\theta} \tag{3.17}$$

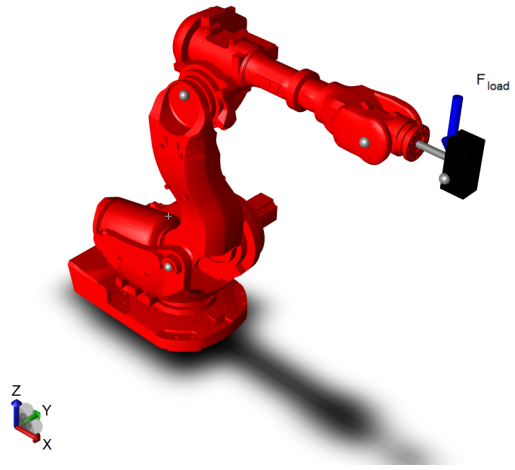
where:

L_1	–	Moment arm of the load applied
L_2	–	Distance from joint 3 to reflector position A
R_4	–	Radius from origin joint 4 to reflector position B
x, y, z	–	Reflector position measured by laser tracker
x_0, y_0, z_0	–	Reflector position measured by laser tracker (unloaded robot)

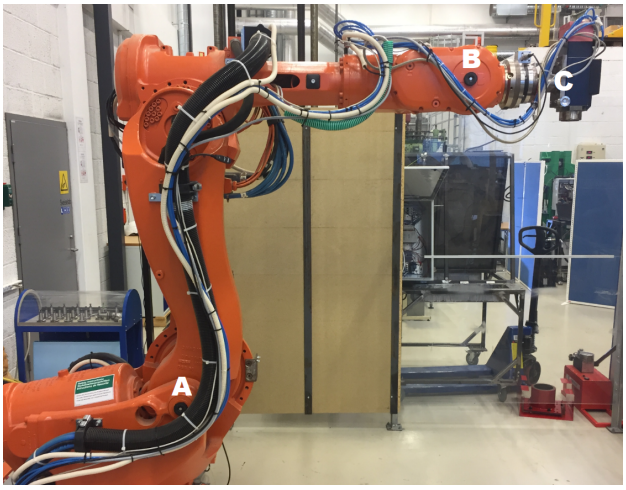
3.1.5 Joint 5

To identify the stiffness in joint 5, the approach is to measure the local angular deflection in the joint. The robot is in home position. Three reflector pucks are placed on the robot, A, B and C, as shown in figure 3.8b. Positions are logged with the reflector attached to all three pucks, with the robot both unloaded and loaded, using the laser tracker. A force is applied to the end-effector as shown with the blue arrow in figure 3.8a. The applied force is logged with the ATI Omega 160. In addition, to cause an angular deflection in joint 5, the applied load

will also cause a small angular deflection in joint 2 and 3. The displacement contribution from joint 2 and 3, is therefor accounted for in the estimation of joint 5.



(a) Applied Force Identifying Stiffness Joint 5



(b) Reflector Placements on Robot while Identifying Stiffness Joint 5

Figure 3.8: Stiffness Identification Joint 5

To estimate the stiffness in joint 5: First, the vectors from A to B, and from B to C, both for loaded and unloaded robot are found:

$$\boldsymbol{AB}_u = [(B_{x_0} - A_{x_0}) \quad (B_{y_0} - A_{y_0}) \quad (B_{z_0} - A_{z_0})] \tag{3.18}$$

$$\boldsymbol{AB}_l = [(B_x - A_x) \quad (B_y - A_y) \quad (B_z - A_z)] \tag{3.19}$$

$$\boldsymbol{BC}_u = [(C_{x_0} - B_{x_0}) \quad (C_{y_0} - B_{y_0}) \quad (C_{z_0} - B_{z_0})] \tag{3.20}$$

$$\boldsymbol{BC}_l = [(C_x - B_x) \quad (C_y - B_y) \quad (C_z - B_z)] \tag{3.21}$$

$$|\boldsymbol{AB}_u| = \sqrt{(\boldsymbol{AB}_{u,x})^2 + (\boldsymbol{AB}_{u,y})^2 + (\boldsymbol{AB}_{u,z})^2} \tag{3.22}$$

$$|\boldsymbol{AB}_l| = \sqrt{(\boldsymbol{AB}_{l,x})^2 + (\boldsymbol{AB}_{l,y})^2 + (\boldsymbol{AB}_{l,z})^2} \tag{3.23}$$

$$|\boldsymbol{BC}_u| = \sqrt{(\boldsymbol{BC}_{u,x})^2 + (\boldsymbol{BC}_{u,y})^2 + (\boldsymbol{BC}_{u,z})^2} \tag{3.24}$$

$$|\boldsymbol{BC}_l| = \sqrt{(\boldsymbol{BC}_{l,x})^2 + (\boldsymbol{BC}_{l,y})^2 + (\boldsymbol{BC}_{l,z})^2} \tag{3.25}$$

Then the angular difference between $\boldsymbol{AB}_u$ and $\boldsymbol{BC}_u$ can be calculated:

$$\phi_u = cos^{-1}\left(\frac{\boldsymbol{AB}_u \cdot \boldsymbol{BC}_u}{|\boldsymbol{AB}_u| \cdot |\boldsymbol{BC}_u|}\right) \tag{3.26}$$

Next, the angular difference between $\boldsymbol{AB}_l$ and $\boldsymbol{BC}_l$ is calculated:

$$\phi_l = cos^{-1}\left(\frac{\boldsymbol{AB}_l \cdot \boldsymbol{BC}_l}{|\boldsymbol{AB}_l| \cdot |\boldsymbol{BC}_l|}\right) \tag{3.27}$$

The angular displacement in joint 5 as a result of the applied load θ is found:

$$\theta = \phi_l - \phi_u \tag{3.28}$$

Finally the stiffness K_5 can be found:

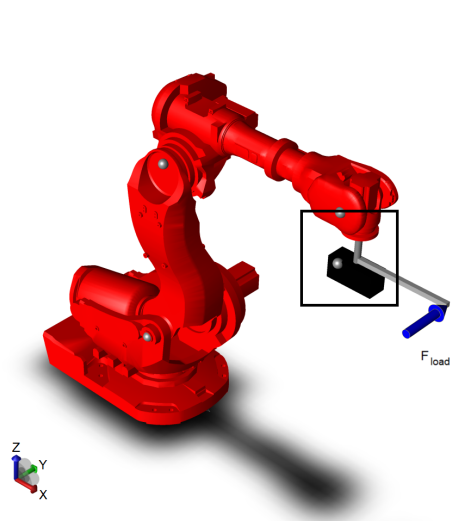
$$K_5 = \frac{F_{load} \cdot L_1}{\theta} \tag{3.29}$$

where:

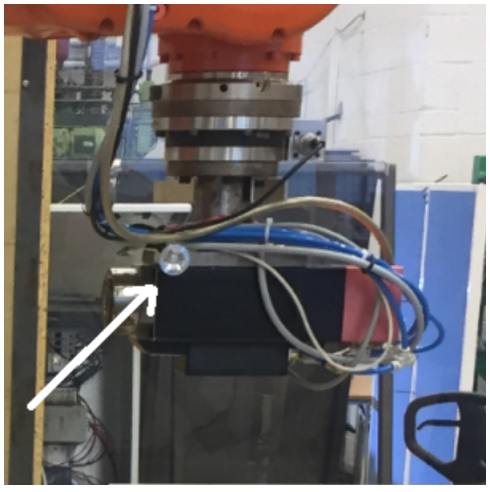
L_1	–	Moment Arm of the Load Applied
x, y, z	–	Reflector Position Measured by Laser Tracker
x_0, y_0, z_0	–	Reflector Position Measured by Laser Tracker (Unloaded Robot)

3.1.6 Joint 6

The approach for identifying the stiffness in joint 6 is a combination of local and global methods. The robot is in home position except from joint 5 which is jogged to $\frac{\pi}{2}$ rad. A reflector is placed on the robot end effector as shown in figure 3.9b, and the reflector position is logged using the laser tracker. A force is applied to the end effector through an arm mounted on the tool as shown with the blue arrow in figure 3.9a. The applied load creates a torque in joint 6, and is logged with the ATI Omega 160. In addition to cause a angular deflection in joint 6, the applied force will also cause a small angular deflection in joint 1. Since the stiffness for joint 1 already is analysed, the measured end effector deflection caused by joint 1 can be calculated and subtracted from the laser tracker measurements.



(a) Applied Force Identifying Stiffness Joint 6



(b) Reflector Placement on Robot while Identifying Stiffness Joint 6

Figure 3.9: Stiffness Identification Joint 6

The joint stiffness in joint 6 is then found:

$$K_6 = \frac{F_{load} \cdot L_1 \cdot L_2}{L} \tag{3.30}$$

$$L = \sqrt{(x - x_0 - d_{1,x})^2 + (y - y_0 - d_{1,y})^2 + (z - z_0 - d_{1,z})^2} \tag{3.31}$$

$$\boldsymbol{d_1} = \boldsymbol{J} * [\frac{\tau_1}{K_1} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T \tag{3.32}$$

$$\boldsymbol{\tau} = \boldsymbol{J} * \boldsymbol{W} \tag{3.33}$$

where:

L_1	–	Moment Arm of the Load Applied
L_2	–	Distance from Joint 6 to Reflector Position (Radius)
x, y, z	–	Reflector Position Measured by Laser Tracker
x_0, y_0, z_0	–	Reflector Position Measured by Laser Tracker (unloaded robot)
$\boldsymbol{d}_1$	–	Displacement Caused by Joint 1
$\boldsymbol{\tau}$	–	Torque Vector
$\boldsymbol{J}$	–	Jacobi Matrix
$\boldsymbol{W}$	–	Wrench from ATI Omega 160

3.2 Simulations

To check and validate the results from the joint stiffness analysis, a model of the robot was developed using the SimulationX software. Multi Body System Mechanics elements were used to build the model. The joints are modelled using actuated revolute joint elements, where a spring-damper element is used to implement the joint stiffness. To avoid oscillations, and ensure steady state, damping is included in the model. For each experiment, corresponding load, including both force and torque, is applied using the global force/torque interfaces with corresponding signal and function blocks. The pre-set element together with the initial function blocks are used to change the robot configuration. CAD files are imported to get correct mass of the robot links, and to ease the visualization of applied forces and reflector placement. An overview of the simulation model is given in figure 3.10.

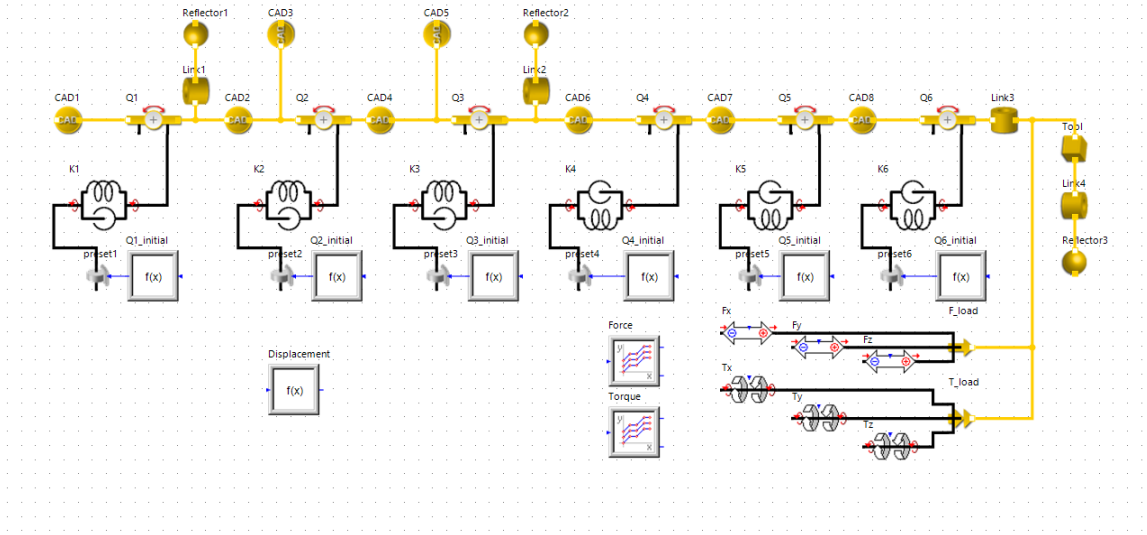


Figure 3.10: Modeling of IRB 6600 in SimulationX

Figure 3.11, show how the robot looks in SimulationX, with the imported CAD files. The blue arrow represents the load from the simulation for joint 1.

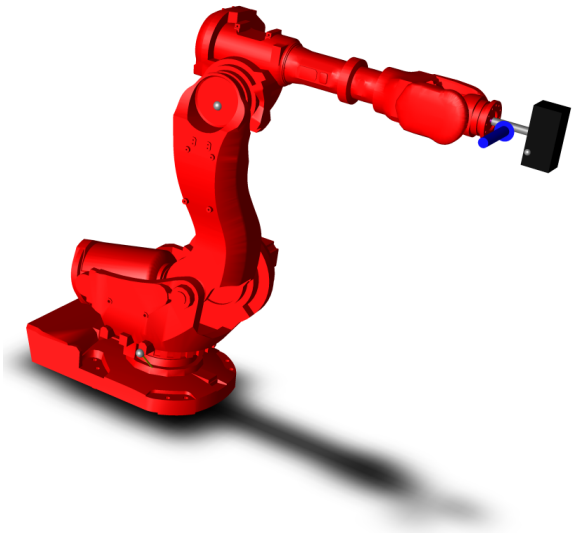


Figure 3.11: Visualization of IRB 6600 with Imported CAD-files

3.3 Pre Loading of Joints

To evaluate if pre-loading a joint have any influence on the joint stiffness, the method described in 3.1 for identification of the stiffness in joint 1 was repeated. To make the joint either pre-loaded or not, the joint was rotated into position from both directions. When the joint is jogged from the side that the load would be acting, the joint is considered pre-loaded. When jogged in the same direction that the load would be acting, the joint is considered not pre-loaded. The

reason for this technique is to eliminate the influence of back-lash in the test. If the stiffens is tested in the backlash region, where the gears are not mated, one estimates the stiffness of the friction in the system, and not the gears. If a larger external force were to be used in the testing, the force would itself force the gears to be pre-loaded.

3.4 Influence of Complementary Stiffness Matrix

The Cartesian stiffness matrix ($\boldsymbol{K}_x$) depends on both the complementary stiffness matrix ($\boldsymbol{K}_c$) and the joint stiffness matrix ($\boldsymbol{K}_\theta$). This is seen in equation 2.39. One approach used in several proposed stiffness analysis methods, like [6] and [9], is to neglect $\boldsymbol{K}_x$. This simplifies the stiffness model of the robot, as only the joint stiffness identification is necessary. In order to investigate the influence of $\boldsymbol{K}_c$ on $\boldsymbol{K}_x$, and to determine if $\boldsymbol{K}_c$ can be neglected in this project, a method from [9] is exploited. The difference in displacement of the robot end-effector pose, when $\boldsymbol{K}_c$ is null and not null, is investigated for different configurations of the robot when subjected to a load.

Two ratios ν_p and ν_r are defined, and used to analyse the influence. ν_p and ν_r are ratios of how the complementary stiffness matrix ($\boldsymbol{K}_c$) influence the position and rotation displacement, respectively. All displacements are found using the equations 2.35 and 2.39.

$$\nu_p = \frac{|\delta p_{\boldsymbol{K}_c} - \delta p_{\boldsymbol{K}_c 0}|}{max(\delta p_{\boldsymbol{K}_c}, \delta p_{\boldsymbol{K}_c 0})} \tag{3.34}$$

$$\nu_r = max\left\{|\delta r_{x\boldsymbol{K}_c} - \delta r_{x\boldsymbol{K}_c 0}|, |\delta r_{y\boldsymbol{K}_c} - \delta r_{y\boldsymbol{K}_c 0}|, |\delta r_{z\boldsymbol{K}_c} - \delta r_{z\boldsymbol{K}_c 0}|\right\} \tag{3.35}$$

where:

$\delta p_{\boldsymbol{K}_c}$	–	Displacement of the Robot End-effector
$\delta p_{\boldsymbol{K}_c 0}$	–	Displacement of the Robot End-effector, when $\boldsymbol{K}_c$ is null
$\delta r_{x\boldsymbol{K}_c}, \delta r_{y\boldsymbol{K}_c}, \delta r_{z\boldsymbol{K}_c}$	–	Angular Displacement (Ang Disp) of End-effector about x_0,y_0 and z_0
$\delta r_{x\boldsymbol{K}_c 0}, \delta r_{y\boldsymbol{K}_c 0}, \delta r_{z\boldsymbol{K}_c 0}$	–	Ang Disp of End-effector about x_0,y_0 and z_0, when $\boldsymbol{K}_c$ is null

As joint 2 and 3 are the joints that contribute most to translational motion of the end-effector, these joints are variables in the calculation. Joint 1 has no influence, and is set to 0. Joint 4,5 and 6, contribute primarily to the end effector orientation, and is set to pi/4 radians, (45°), in order to keep the wrist configuration far from singularities [9]. Results from the performed joint stiffness analysis are used to generate $\boldsymbol{K}_\theta$. Joint 2 ranges from -65° to 85° and joint 3 ranges from -180° to 70° The wrench, containing forces and moments $\boldsymbol{W}_{load}$ is set to $[0N\ 0N\ -2500N\ 250Nm\ 250Nm\ 0Nm]$, which is higher than the process forces logged during milling, but considered as a reasonable load for the IRB6600 robot. The MATLAB®script used to calculate the influence is found in appendix A.14.

3.5 Kinematic Performance

One way to evaluate the kinematic performance is by the Jacobian condition number. The condition number indicates the invertibility of the Jacobian matrix, and also how far a robot configuration is from singularities [14].

There are several different normalization methods that can be used to calculate the condition number, but only the Frobenius norm is frame-invariant, meaning that it is dependant of q_1, and considered an analytic function [14]. Thus, the Frobenius norm can provide an expression for the condition number, based on different joint configurations. This is an advantage compared to using,e.g the 2-norm, which only can calculate singular values [9]. The Frobenius norm of an mxn matrix $\boldsymbol{A}$ (with $m, n \geq 2$) is defined in [15] as:

$$||\boldsymbol{A}||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} \tag{3.36}$$

For a robot which can both move and rotate the end effector, not all the Jacobian matrix entries will have the same units, and calculation of the condition number will not make any sense [14]. Therefore, the characteristic length of the robot L_c is introduced to normalize the Jacobian matrix, making it homogeneous [16].

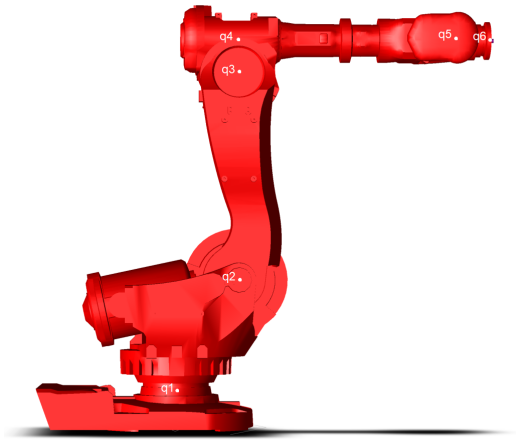


Figure 3.12: IRB6600 in Home Position

Figure 3.12 show the ABB IRB6600/175-2.55 in home position, and all the joints, $q_1 ... q_6$, are marked. The characteristic length L_c, is found as follows:

$$L_c = \frac{R_r}{R_m} \tag{3.37}$$

$$R_m = \boldsymbol{p}^1_{6_{max}} = \boldsymbol{p}^1_2 + \boldsymbol{p}^2_3 + \boldsymbol{p}^3_5 + \boldsymbol{p}^5_6 \tag{3.38}$$

where:

$$\begin{aligned} R_r &- \text{Reach of Robot (Found in Datasheet [1])} \\ R_m &- \text{Maximum Reach of Robot} \\ \boldsymbol{p}_j^i &- \text{Vector from Joint } i \text{ to joint } j \end{aligned}$$

The normalized Jacobian matrix $\boldsymbol{J}_n$, can then be determined:

$$\boldsymbol{J}_n = \begin{bmatrix} \frac{1}{L_c}\boldsymbol{I}_{3x3} & \boldsymbol{0}_{3x3} \\ \boldsymbol{0}_{3x3} & \boldsymbol{I}_{3x3} \end{bmatrix} \boldsymbol{J} \tag{3.39}$$

The Frobenius norm of the normalized Jacobian matrix is then calculated. Joint 2 and 3 are the joints that contributes most to translational motion of the end-effector, these joints are therefore variables in the calculation. Joint 1 has no influence, and is set to 0. Joint 4,5 and 6, contribute primarily to the end-effector orientation, and is set to pi/4 radians, (45°), in order to keep the wrist configuration far from singularities [9]. Joint 2 ranges from -65° to 85° and joint 3 ranges from -180° to 70° The MATLAB®script used in the calculation is found in the appendix A.15.

3.6 Region of Operation

To determine the region of operation for the robot, i.e. positioning of the work object related to the robot, several elements had to be considered. Figure 3.13, illustrates the orientation of the door in relation to the robot.

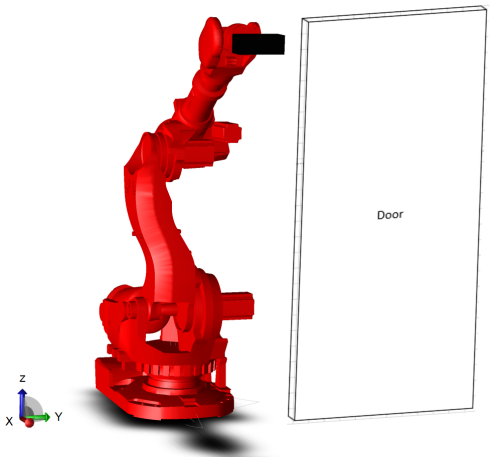


Figure 3.13: Robot with Door

The different door sizes given in the requirements are used to calculate the work space which the robot has to be able to cover:

$$\delta x = T_d \tag{3.40}$$

$$\delta y = \frac{W_{d,max}}{2} - \frac{W_{d,min}}{2} \tag{3.41}$$

$$\delta z = H_d - 2 \cdot \delta H_{end} \tag{3.42}$$

where:

$\delta x, \delta y, \delta z$	–	Translational Movement in X,Y and Z
T_d	–	Door Thickness
W_d	–	Door Width
H_d	–	Door Height
δH_{end}	–	Minimum Length from Bottom of Door to Hinge

This work area defined by $\delta x, \delta y$ and δz, is then used together with calculated results regarding kinematic performance, and the inverse kinematics, to find a proper region of operation for the robot.

3.7 Door Localization

The milling process starts with a single door being placed in the milling fixture; with a given tolerance from origin. It is crucial for the milling application to know the door position with high accuracy. Regarding the milling task, a highly accurate method to locate the door is highly necessary to ensure the best possible product. There are many different methods to achieve the measurements of the door, and they will be discussed in this section, and validated using the following criteria.

- Cycle Time
- Price
- Accuracy
- Durability

The placement method is allowed to have some deviation from the desired accuracy of $\pm 0.5mm$. If the door was to be placed with high accuracy, the placement system would be expensive due to high accuracy requirements. Accuracy is critical in milling applications, to ensure that the tool is engaging the work-piece in a desired manner. The door localization system is therefore given the following maximum deviations to compensate for:

- Vertical Offset: $\pm 10mm$
- Horizontal Offset: $\pm 10mm$

- Angular Offset: $\pm 2°$
- Door Thickness: $30 - 100mm$
- Door Width: $290 - 1350mm$
- Door Height: $1200 - 3000mm$

Probing

The method which requires the least computational power is traditional probing. This is achieved by installing a sensing tool in the robot's end effector. This can either be a digital or an analogue sensor, where the analogue sensor will allow for a smoother measurement cycle for the robot (no sudden stopping). The process is executed in the following manner: The robot approaches the door on multiple probing points, to determine the pose of the door, such that a work-object coordinate system can be generated. This method is inexpensive as there is no need for an industrial computer, nor highly accurate sensor-mounting fixtures are needed. It is accurate due to the sensor being placed in the robot's tool-tip, durable and contains high repeatability (when the robot is often calibrated). However, the method requires physical movement of the robot, which takes a substantial amount of time compared to the other considered methods. When using robots in assembly lines, low cycle time is of high importance. Hence, the time needed for the movement of the robot results in a large drawback for this solution. The overall score can be seen in table 3.1, and the solution can be seen in figure3.14.

Table 3.1: Score: Probing

	Time	**Price**	**Accuracy**	**Sum**
Score[1-6]	1	6	5	12

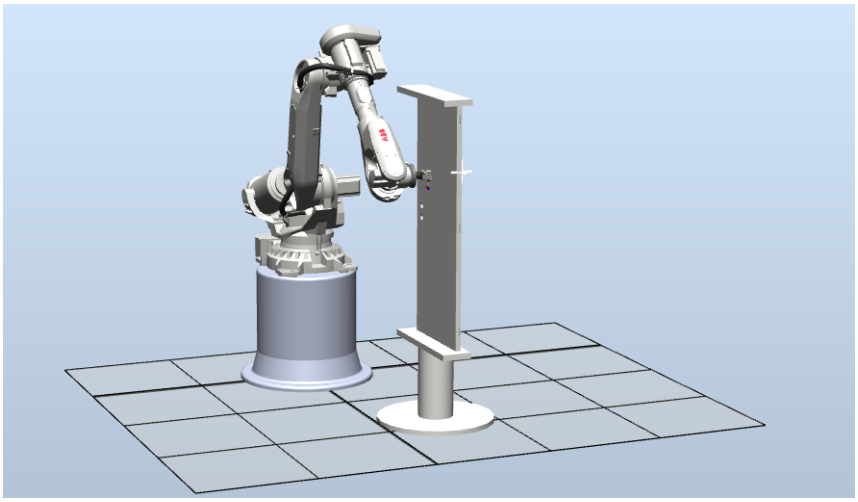
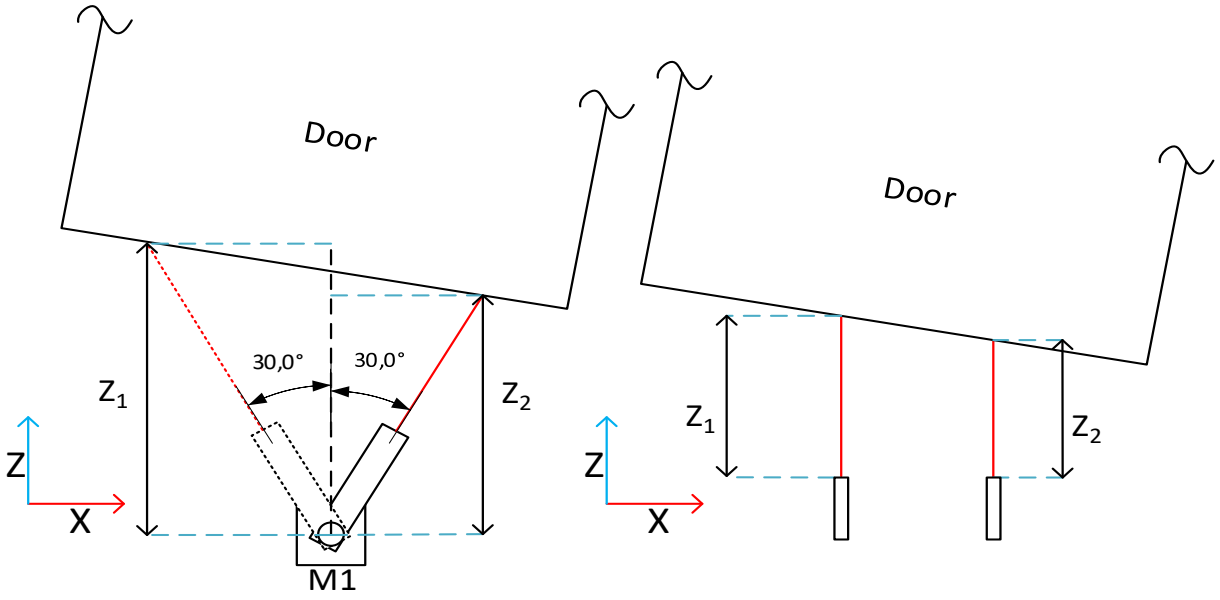


Figure 3.14: Probing Cycle

Photoelectric Sensor

By using a total of six high accuracy photoelectric sensors, the door's pose can be located in space with high accuracy. This is achieved by using the known geometry of the door to calculate the pose. It should be noted; that to achieve the highest accuracy from the sensors, they should be placed as far away from one another as possible. Due to the high cost of the sensor, a solution where some of the modules are mounted on a high accuracy motor will be discussed. Here, the motor is positioning the laser at two different known angles with respect to the door. By knowing the laser angles, the vertical/horizontal distance can be calculated, and a reduced number of sensors will yield the same results. An open-loop controlled stepper motor can be used to achieve high accuracy with an inexpensive set-up. The difference between the two solutions can be seen in figure3.15a and figure3.15b, respectively.



(a) Motor Configuration (b) Dual Sensor Configuration

Figure 3.15: Probing to Determine Door Pose

The extracted information can further be used to define the pose of the door, which yields its coordinate system. The price of a laser is highly determined by its working range and resolution. Thus, short working range is desirable. By using the set-up as seen in figure 3.16 (where laser 2,3 and 4,5 can be reduced to only 2 lasers by using a motor) the need for a laser with high working range is reduced to only one. The other 5 lasers are determining the pose of the door, except the x position. By knowing the geometry of the door, and exact position of the lasers, this value can be calculated. With the configuration seen in figure.3.16, it is clear, that laser 1 needs a greater work range in contrast to the other 5.

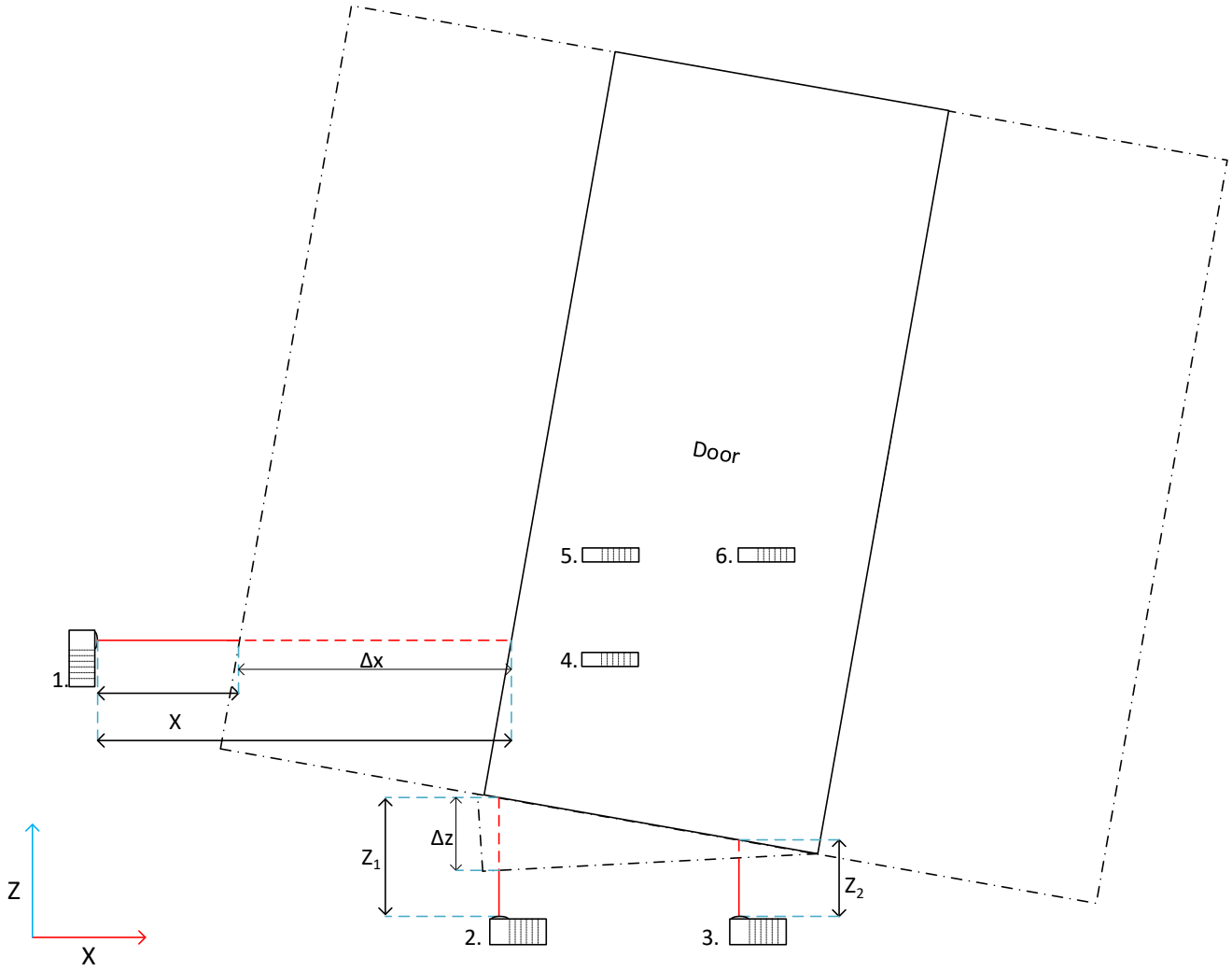


Figure 3.16: 6-DOF Photoelectric Sensor Configuration

The main difference with photoelectric sensors, compared to the previously mentioned sensor method is its lack of physical movement. The photoelectric sensors set-up is effective due to its high resolution, high repeatability and durability, and most important, very low cycle time. By taking example in the OPT2002/BOD0025[A.21][A.20] sensors, the resolution can be as good as $8\mu m$ which is well within the requirements of $500\mu m$. It should be noted; that these values are valid on optimal measurements surfaces with parallel faces to the laser. The accuracy is reduced by offset angles and highly reflective surfaces. However, the system has a higher price then the probing system, and it needs some housing to avoid clogging from dust and particles. The overall score of the laser solution can be seen in table.3.2.

Table 3.2: Score: Laser Probing

	Time	**Price**	**Accuracy**	**Sum**
Score[1-6]	6	2	6	14

Camera

Camera is also an available alternative concerning the pose determination of the door. A camera solution is inexpensive, but it requires much computational power, and it is sensitive to dust and particles. When using cameras, the camera data is fed into recognition software, this could be a source which introduces uncertainties. A camera has a given maximum resolution, it is important to select a camera with sufficient resolution to the given application in order to achieve the desired accuracy. Regarding cameras, there are two opportunities for the configuration and method used. One can use stereo vision in two planes, or (to achieve a higher resolution and less computational power) the cameras can be placed in three different axes to determine the pose of the door. Both solutions are discussed in this section.

Axis Vision

To determine the pose of the door, three cameras can be mounted as seen in figure 3.17.

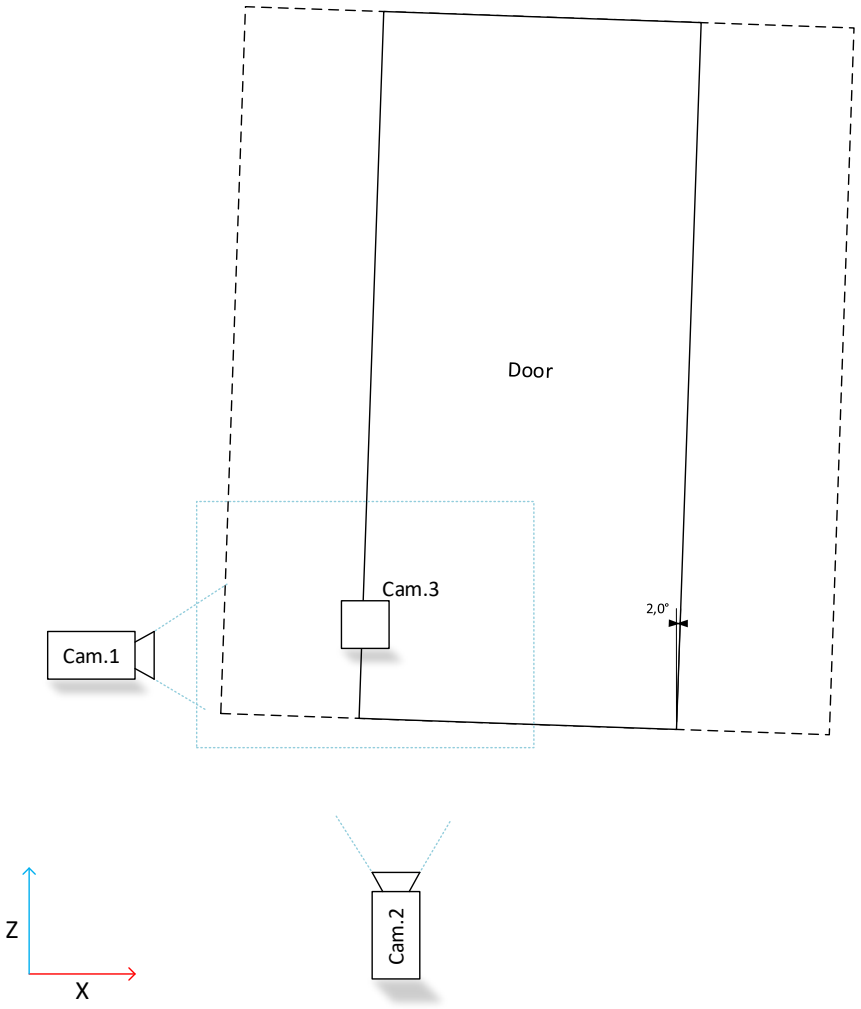


Figure 3.17: 6-DOF Axis Vision System

Camera 1 is responsible for deviation in z- and y-axis as well as rotation around the x-axis. Camera 2 determines x position and the rotation about the z-axis and camera 3 determines the rotation about the y-axis. Extracting this information is fundamental to determine the pose of the door. Further, for some points, multiple cameras will be able to retrieve the same information. In these cases the information will be merged to achieve the highest possible accuracy. Due to the depth of the door, camera 2 and 3 cannot be positioned directly in-line with the door sides. If this should occur, the backside of the door can be seen in the frame if any door rotation occurs, which again can create difficulties for the corner detection software. The door width can also vary, with a known length, so to position the cameras in the measuring location, stepper-motor driven sliders might have to be installed, as seen in figure.3.18.

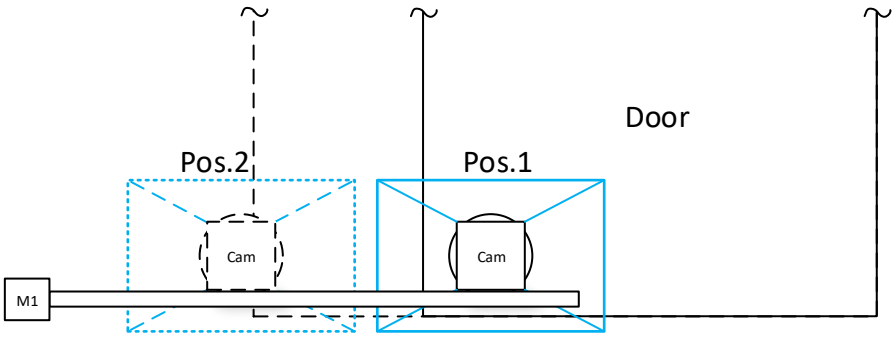


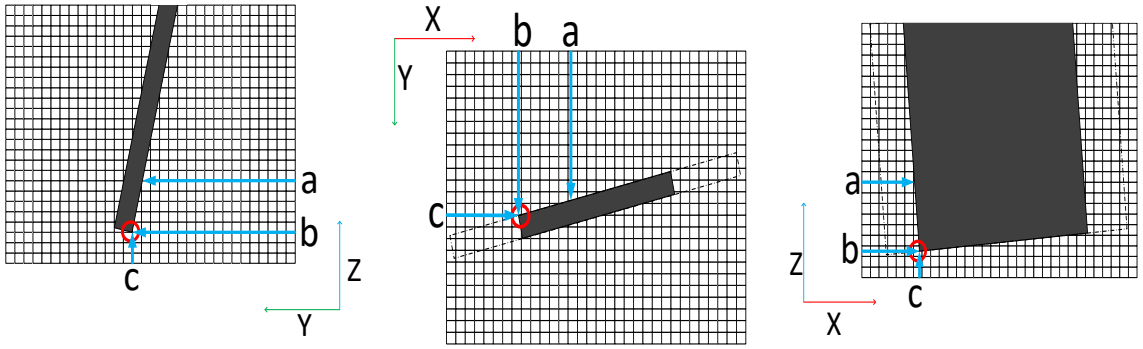
Figure 3.18: Stepper Drive Camera System

The resolution of the measurements is determined by the following factors: Camera to door distance, camera resolution and processing code and software. In theory, with a 1080p camera which has a resolution of 1920x1080 pixels, it should be able to determine the corner of a door with 0.1mm accuracy, if the camera sees a grid of 192x108mm. However, installing a camera with such accuracy is not realistic, a calibration is therefore needed. The calibration method is as follows (For camera 1.):

- Position door in origin (Wanted position)
- Subtract the pixel values in both Y and Z direction (Make sure door rotation is 0°)
- Move the door positive 10mm in both directions. (The deviation in pixels from the origin should be the same in both axes due to the uniform distribution of pixels)
- Obtain the new pixel values and calculate scaling factor by: $\frac{10mm}{\text{pixel deviation}}$
- With calculated scaling-factor, one can determine the offset by multiplying the off-set pixel value with the scaling factor to determine off-set in mm.

It is known, that the sizes of the doors vary. This fact creates the need for a dynamic scaling factor, due to the door being closer to the camera. This is done by using the door width as

the only known parameter, and by knowing the camera location, the scaling factor can be dynamically calculated for each door. This technique is only necessary for camera one, due to the distance deviation for the other two can be neglected. The grid seen by cameras 1, 2 and 3 are shown in figure 3.19a, 3.19b and 3.19c respectively. (However, this approach is only necessary if the slider method is not used.)



(a) Camera 1 (b) Camera 2 (c) Camera 3

Figure 3.19: Camera Grids for Camera 1,2 and 3.

As seen in the figure above, the b and c arrows determine the location of the coordinate system origin, whereas a and b is used to calculate its receptive angle.

To thoroughly investigate this solution, a test-rig was designed as shown in figure 3.20. The test-rig used the FOV of camera 1, and served as a proof of concept for this solution. The experiment was programmed in NI LabVIEW, and the output was a *.txt* file with sufficient information to define a door origin. The vision code can be seen in appendix A.19 The code executes the following tasks.

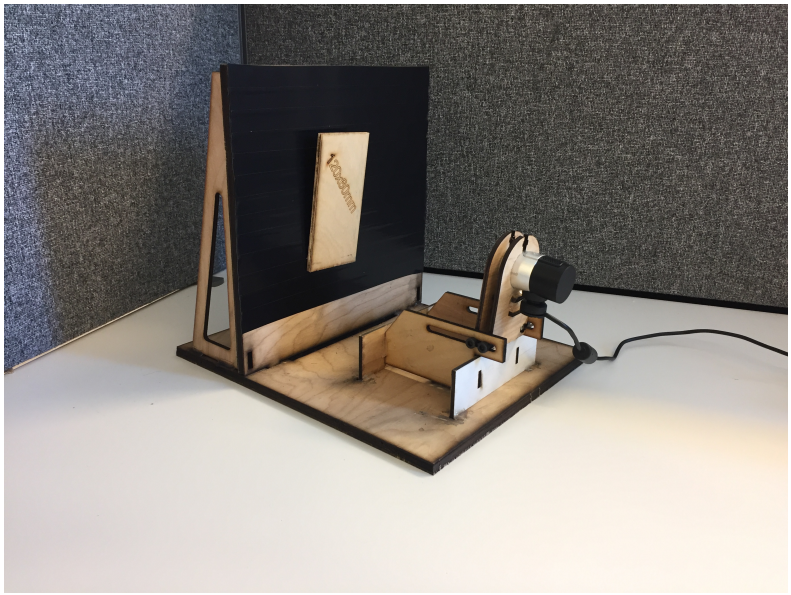


Figure 3.20: Axis Vision Jig

1. Extract luminance plane
2. Find line in X-direction
3. Find line in Z-direction
4. Set coordinate system in intersection, with angle based on the line in X-direction.

Table 3.3: Score: Axis Camera Probing

	Time	**Price**	**Accuracy**	**Sum**
Score [1-6]	5	5	5	15

Stereo Vision

The dual stereo vision solution differs from the previous solutions due to its ability to determine depth in an image. However, for locating pose with high accuracy and repeatability with uniform surfaces, this solution will require excessive computational power and add a high level of uncertainty. The main problem that makes the stereo short coming is the need for a door angle. To extract the angle using stereo vision, two known points must be recognized in both the images, this is problematic, due to the doors being single colour. This problem can be solved by installing cameras so that both the bottom corners can be observed, as shown in figure 3.21. However, this result in a lager field of view, which spread the pixels over a maximum horizontal line at $1,35m$ (door maximum width) + baseline, and with 1980 pixels in the horizontal plane, the resulting resolution will exceed 0.5mm.

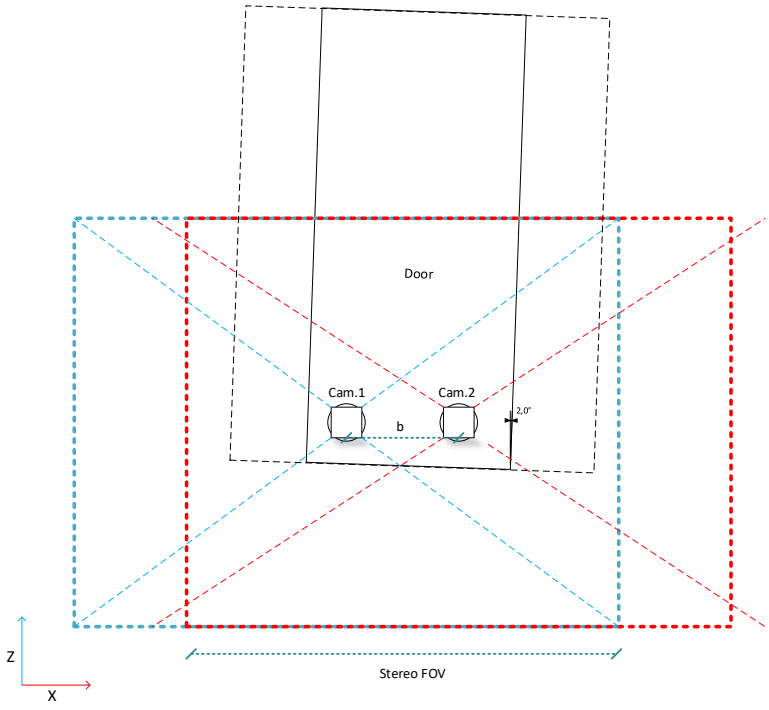


Figure 3.21: Stereo Vision Configuration

Camera Calibration

As described in section 2.3, the MATLAB®camera calibration toolbox is used to calibrate the camera in this project. The chequerboard shown in figure 3.22 is placed on a flat surface and 20 images are captured from different distances and angles. This is illustrated in figure 3.23a. The camera resolution is set to 1920x1080.

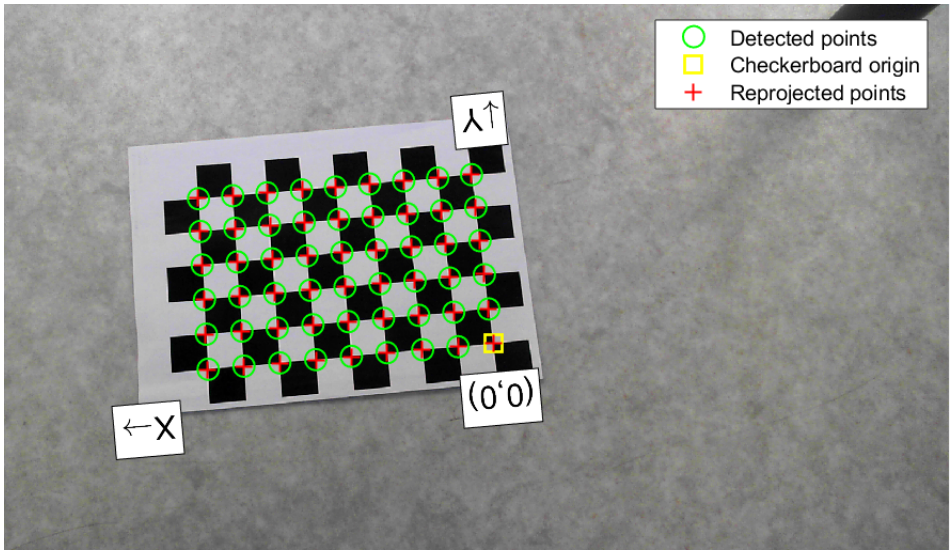
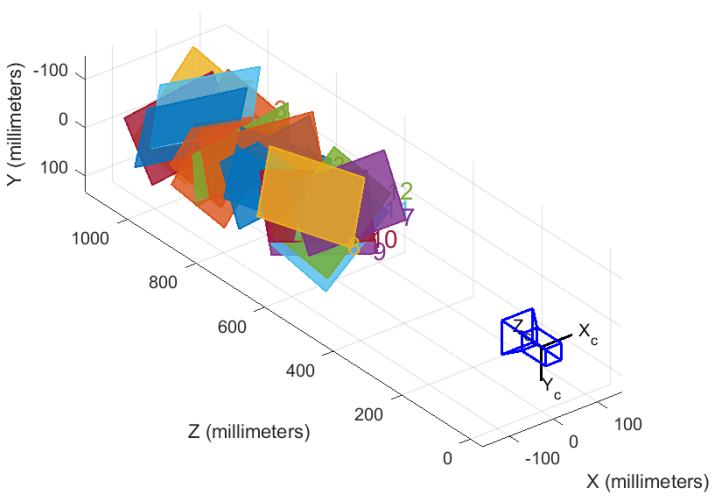
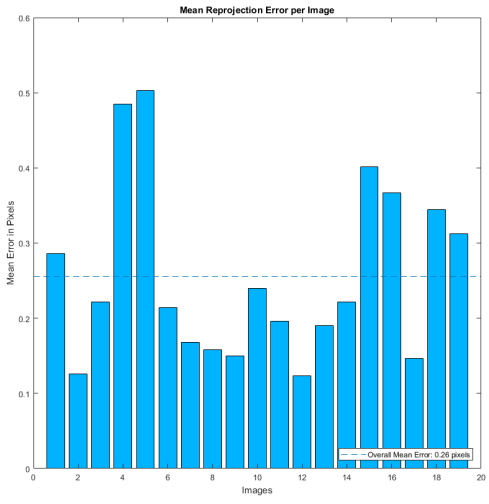


Figure 3.22: Checkerboard used for Camera Calibration

One of the captured images were rejected due to high pixel error. The mean pixel error of the 19 images that were approved, is shown in figure 3.23b.



(a) Visualization of the Captured Images



(b) Mean Reprojection Error per Image

Figure 3.23: From MATLAB®Camera Toolbox

The results from this camera calibration are stored in the $\boldsymbol{k}$ matrix which hold the camera

intrinsic parameters, and the distortion factors.

$$\boldsymbol{k} = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1678.35 & 0 & 946.41 \\ 0 & 1727.22 & 510.71 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.43}$$

where:

$$\begin{aligned} f &- \text{Focal lengths in pixels} \\ C &- \text{Principal point position in pixels} \end{aligned}$$

$$\text{Distortion coefficients} - [0.0206 \quad 10.4073 \quad -115.4536]$$

Both the intrinsic parameters in the $\boldsymbol{k}$ matrix, and the distortion coefficients are implemented in the vision algorithm used for door localization.

3.8 Automatic Generation of Robot Trajectory

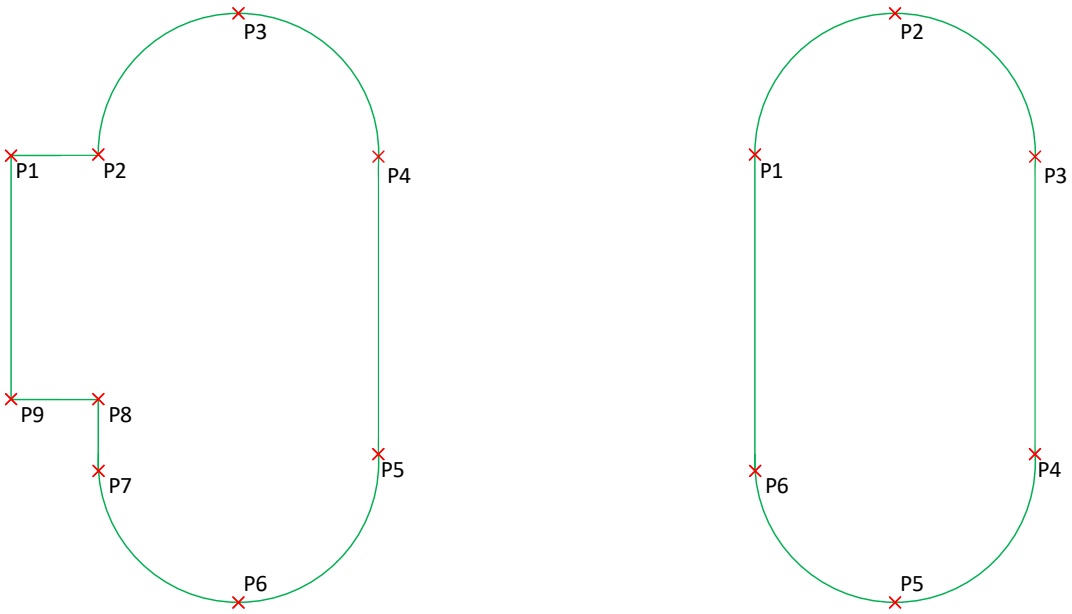
Throughout this section, the entire process from a door presented in its clamping-rig, to a finished milled product will be covered. Experiments have been conducted concerning the different modules of the process to ensure the durability and overall functionality of the path generation. A proof of concept approach was used to reduce the work-load, while exhibiting the system capabilities. The system could be modified to mill both lock-case and handle features, but throughout this project, the focus has regarded the hinges.

When visiting the door manufacturer, it was explained that the door would only be displaced in space in 3 degrees of freedom X, Z and roll. It was decided to use the probing method with camera, as seen in section 3.7. Due to the restrictions set by the door manufacturer, one camera on a sliding mount would yield sufficient placement data, in this case, camera 3. To achieve a fully automated process, different software is used such as MATLAB®, NI LabVIEW and ABB Robot Studio.

Feature Data

The feature and door data (FDD) are presented to the Path Generation system (PGS) in form of an excel file. The data listed in the file is sufficient to mill the desired features in the door. Here, also the force displacement is listed. Further, the milling path consists of 9 points for each hinge path, both y- and z-axis compensation are added for these points. The path with its targets can be seen in figure 3.24a. However, only a simple fillet with 6 points is used throughout the path generation to allow for the best possible force readings. The used path, with its respective points can be seen in figure 3.24b. The data in the file consists of:

- Door Width
- Door Height
- Door Thickness
- Number of Hinges
- Hinge Position
- Hinge Force Displacement
- Hinge Geometry
- Tooling Speed and Feed-Rate



(a) 9 Hinge Targets

(b) 6 Hinge Targets

Figure 3.24: Comparison between 6 and 9 Hinge Points

By using the sensor system in combination with the FDD information, the PGS can utilize the known trigonometry to calculate the desired robot target to account for 3 DOF displacement; X, Z and Yaw, as seen in figure 3.25. New doors can be easily added to the library by using such a method. The simplicity in this process is highly important due to the fact that the door manufacturer only designs tailored products.

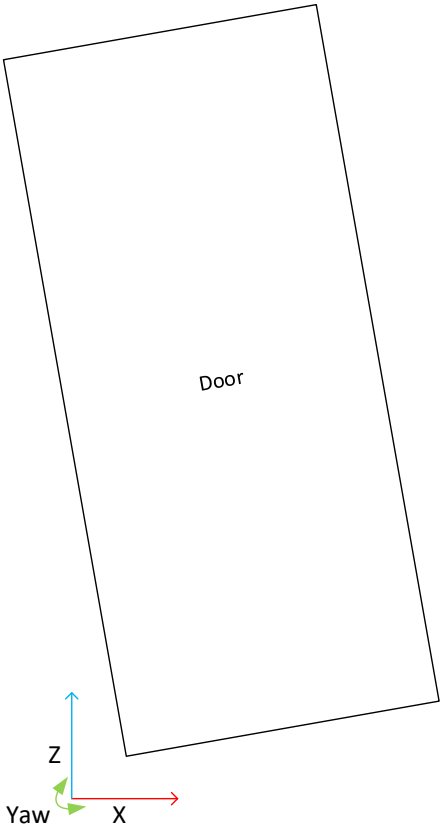


Figure 3.25: Allowable Degrees of Freedom

Path Generation

The path is deigned in NI LabVIEW with multiple sequences. First, the initial steps such as feature extraction, angle conversion, quaternion data string conversion and sensor data gathering will be covered. It is important that these steps are executed initially seeing that this data is vital for further code and calculations. In the next sequence the desired coordinates are calculated using the previous gathered data. Next, the data is converted from float values to strings, so that the values can be further read by the RAPID program. In the final step, the desired data file is opened, and the generated data is written to the file. All the listed sequences will be described in this section.

Feature Extraction

To extract the FDD from the excel file with ease, the excel file is saved as a .txt file. The data is extracted from the file and saved as integer values in their respective variables. This process is achieved by accessing the desired rows and columns where the desired data is stored. The code can be seen in figure.3.26.

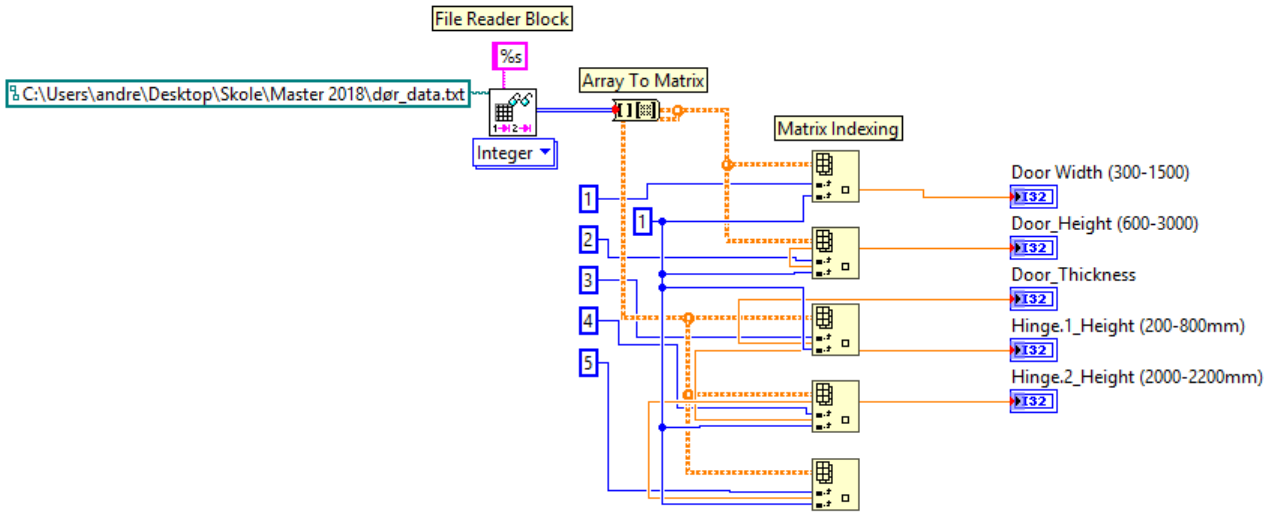


Figure 3.26: Feature Extraction Code

Angle To Quaternion Conversion

The angular displacement of the door is represented by a quaternion in RAPID code, but it is extracted as angles. The door data is therefore initially converted to radians, and further to quaternions. Due to the door rotation being restricted to rotate only about one axis, only two of the four normalized quaternion values are altered. The conversion can be seen in figure 3.27.

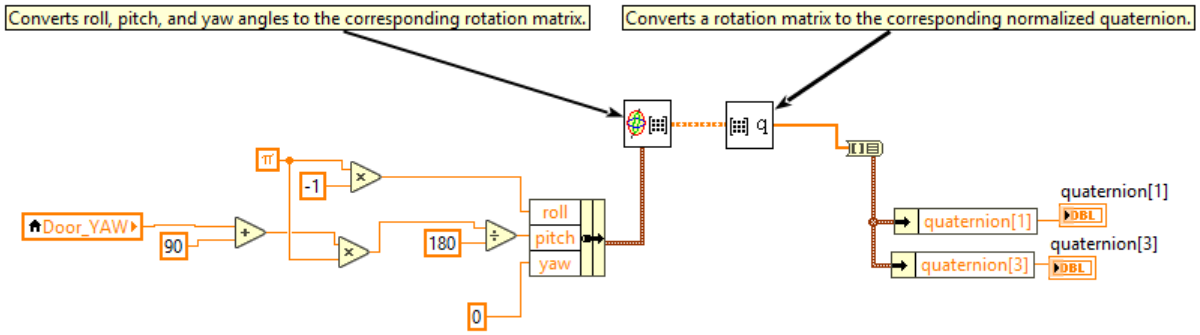


Figure 3.27: Angle to Normalized Quaternion Conversion

Door Localization

The sensor data is gathered by using techniques presented in section 3.7. The system is responsible for detecting the origin displacement of the door in both x- and z- axes and the angular displacement. As mentioned in section 3.7, it can be seen that the calibration data is here added between the "Vision Acquisition" and the “Vision Assistant” block. This block ensures that the

retrieved picture represent the real-world in a improved manner. The "Vision Assistant" block outputs the doors origin position in pixels (relative to the cameras origin (Bottom left corner)), the data is further offset in both x- and z-axes to offset the data from the origin. Further, the data is multiplied with a conversion multiplier to extract the metric displacement of the door in x- and z-axes and angular displacement in degrees. The code used to determine the door position is shown in figure 3.28.

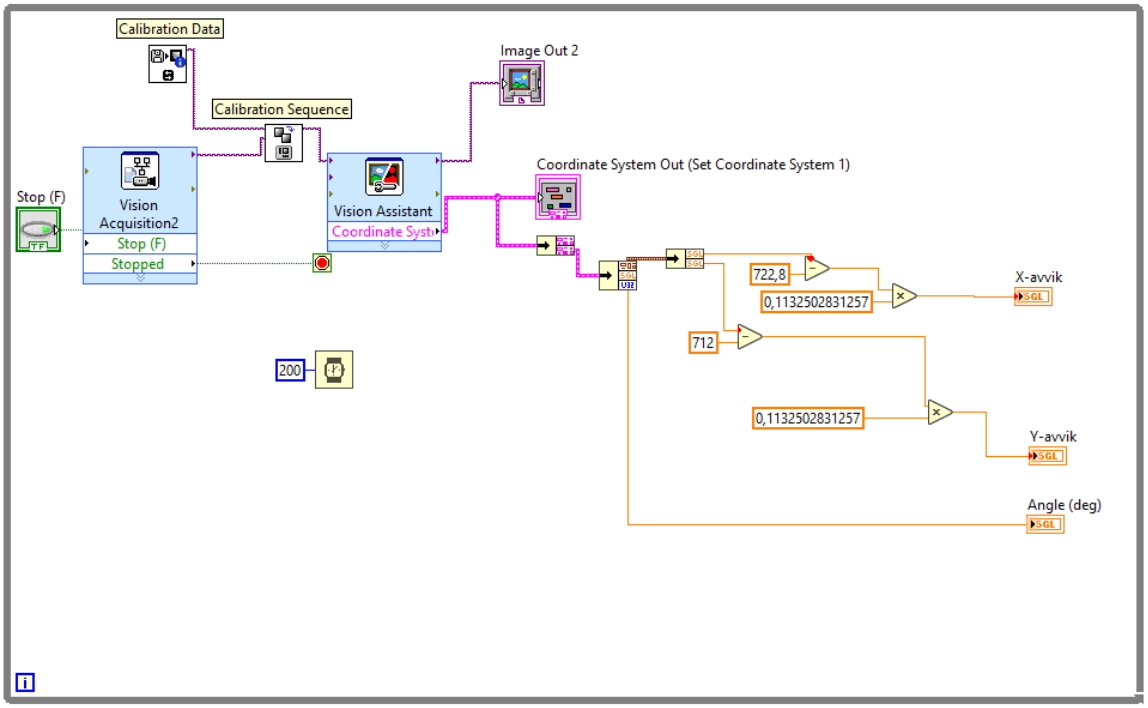


Figure 3.28: Vision System

Process Force Correction

The off-line correction data is, as mentioned in section 2.4, a product of robot configuration, stiffness and external forces. The milling forces acting on the tool will always work in the same direction with respect to the spindle head. Consequently, the path correction can be added to the different axis after the final spindle pose has been calculated. The generated forces in the milling process is pushing the entire spindle in the direction of which the forces are acting. This results in either over or under cutting of the nominal path dependent on the milling forces (climb/conventional). The results of an uncompensated milling sequence can be seen figure 3.29, where F_t is the thrust force and F_c is the cutting force [6].

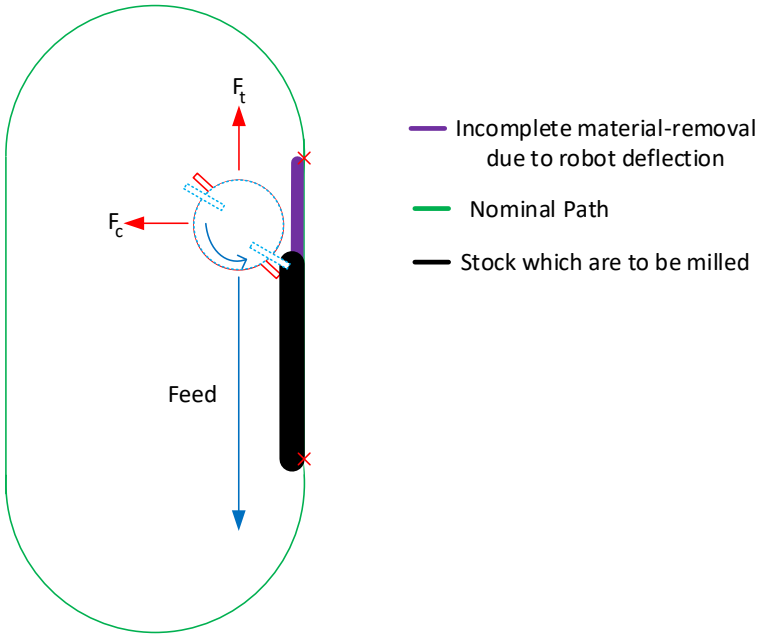


Figure 3.29: Uncompensated Milling

The forces acting on the mill is assumes similar with respect to its direction. Thus, the corrected path would be offset, as seen in figure 3.30. With the corrected path off-set, the desired result is trace the nominal path with as little deviation as possible.

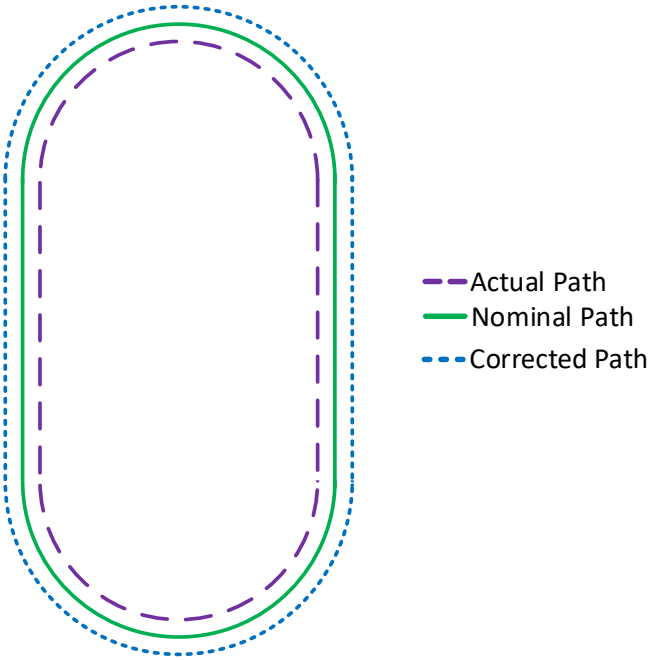


Figure 3.30: Path Comparison

To properly calculate the distance of which the TCP has to be corrected, the individual joint-torques must be identified to compensate the target position. This is achieved by logging the TCP wrench with the ATI 160 during milling. This data is the process forces, $\boldsymbol{W}_p$, where the mean values are calculated. In combination with the Jacobian matrix, they are used to calculate the joint torques as seen in section 2.2:

$$\boldsymbol{\tau} = \boldsymbol{J}^T \cdot \boldsymbol{W}_p$$

When the torques are known, the joint deflection can be found. This step uses the stiffness data from the analysis in section 3.1 to create the diagonal stiffness matrix $\boldsymbol{K_\theta}$. From equation 2.36, in section 2.2, the following equation is derived:

$$\delta \boldsymbol{q} = \boldsymbol{K}_\theta^{-1} \cdot \boldsymbol{\tau} \tag{3.44}$$

Also as shown in section 2.2, the TCP deflection can be calculated as:

$$\delta \boldsymbol{d} = \boldsymbol{J} \cdot \delta \boldsymbol{q} \tag{3.45}$$

The calculated deflection is then corrected for in the path trajectory generation. To properly implement the calculations into the process, it was chosen to add the calculations to the NI LabVIEW code. Seeing that these calculations are needed for each of the points in both the fillets, computational-time exceeding 1s is needed. However, the calculation of the TCP deflection can start as soon as the door parameters are known. Conclusively, the time which is spent picking and placing the door is simultaneously used to calculate both the fillets displacements and add them to the path. The calculations of all the points of both the fillets can be seen in appendix A.16, and the final conversion to x- and z-axis coordinates is presented in figure 3.31.

```
42  Delta_P1_P2_P7_x = D_P1_P2_P7(1);
43  Delta_P1_P2_P7_z = D_P1_P2_P7(3);
44  Delta_P3_x = D_P3(1);
45  Delta_P3_z = D_P3(3);
46  Delta_P4_P5_x = D_P4_P5(1);
47  Delta_P4_P5_z = D_P4_P5(3);
48  Delta_P6_x = D_P6(1);
49  Delta_P6_z =D_P6(3);
50
51
52  Delta_P1_2_P2_2_P7_2_x = D_P1_2_P2_2_P7_2(1);
53  Delta_P1_2_P2_2_P7_2_z = D_P1_2_P2_2_P7_2(3);
54  Delta_P3_2_x = D_P3_2(1);
55  Delta_P3_2_z = D_P3_2(3);
56  Delta_P4_2_P5_2_x = D_P4_2_P5_2(1);
57  Delta_P4_2_P5_2_z = D_P4_2_P5_2(3);
58  Delta_P6_2_x = D_P6_2(1);
59  Delta_P6_2_z =D_P6_2(3);
```

Figure 3.31: Data Correction

Target Generation

When the initial data is gathered, the path of the robot must be generated. To calculate the desired robot targets, the following data must be known relative to the door: target X and

Z position, and the angular displacement relative to the global coordinate system. Both the off-set in X and Z are compensated for in the *robot offset* (shown in figure 3.33), but the angle is needed to calculate the added contribution. The door with its respective variables can be seen in figure 3.32. The equations used to determine the desired point can be seen in equations 3.46- 3.50. The implementation of the equations in the code was achieved using MathScript, as seen in figure.3.33, where three of the fillet points are generated.

$$L_p = \sqrt{{Lx_L}^2 + (Lz_L + PC)^2} \tag{3.46}$$

$$\alpha = \cos^{-1}\left(\frac{Lx_l}{L_p}\right) \tag{3.47}$$

$$\beta = \alpha + Yaw \tag{3.48}$$

$$Lx_G = L_p \cdot \cos\beta \tag{3.49}$$

$$Lz_G = L_p \cdot \sin\beta \tag{3.50}$$

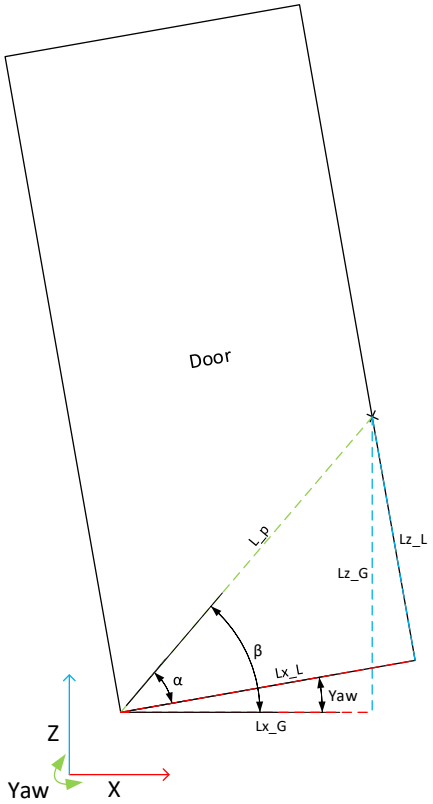


Figure 3.32: Point Generation Trigonometry

where:

α	-	Angle of Local Point
Yaw	-	Angular Displacement
L_p	-	Absolute Distance to Point
Lx_L	-	Local x-distance
Lz_L	-	Local z-distance
PC	-	Point Correction in z-axis for Each Point.
Lx_G	-	Global x-distance
Lz_G	-	Global z-distance

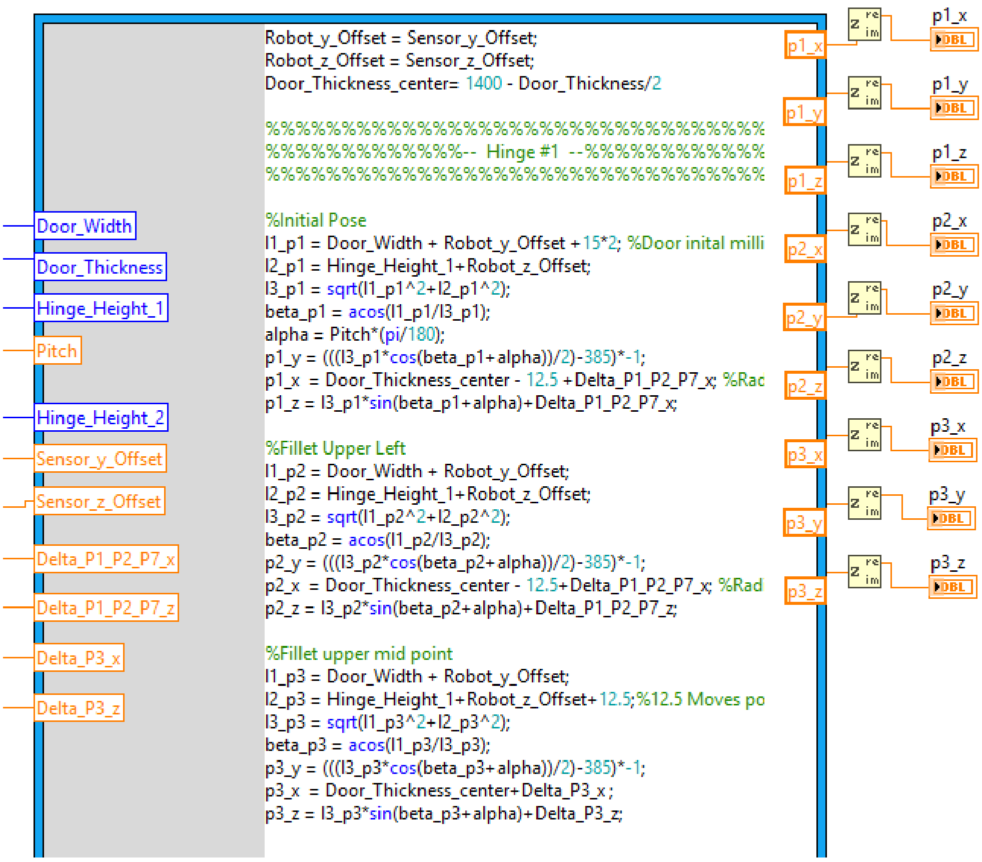


Figure 3.33: Equation Implementation using MathScript

Robot Target Conversion

To convert and merge the generated data into robot targets, strings with "." separated values instead of "," needed to be available. Further, the code also allows for the freedom to choose the desired decimals in the value. To make the information readable for the RAPID-code later on in the process, the data needs to be separated. In this case, the symbol ";" is used. The following code seen in figure 3.34, was used for each individual robot target.

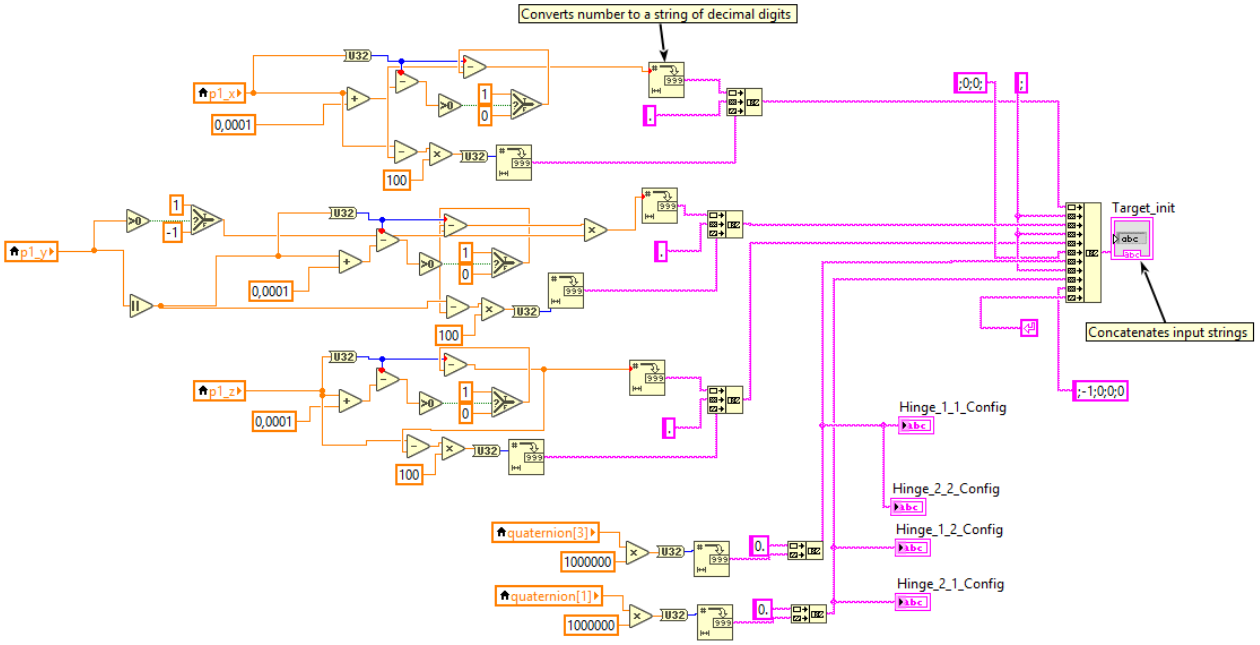


Figure 3.34: Robot Target String Generation

RAPID File Generation

The NI LabVIEW program is dedicated to open the source file, write the generated data, and close the file. By using this code, the entire stage from the door being presented, to generated robot targets have been achieved. The final step can be seen in figure 3.35.

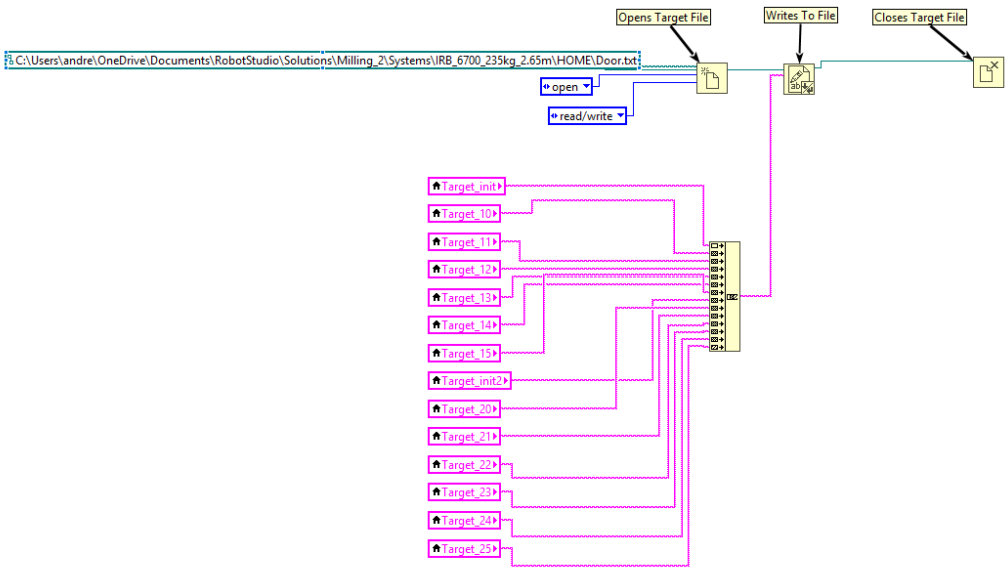


Figure 3.35: RAPID Read File

Path Extraction

The purpose of the automatic target generation is to reduce and streamline the production process. Therefore, the entire process can be started by a simple touch on the Teach Pendant or on a computer. The RAPID-code will then start by setting a signal HIGH, which is further read by the NI LabVIEW code. This signal means that the placement of the door is complete, and the target generation process can initiate. After execution, the NI LabVIEW code resends a signal along with the generated targets. This verification signal is important, to ensure that the correct data is being used to construct the robot targets.

The following steps needed to successfully import a robot target from an external file will be here described. This entire process is needed due to the lack of RAPID functions to upload robot targets directly. First, the variable where the desired data is to be stored needs to be declared. Further, the data needs to be stored in string variables. This is achieved by opening the NI LabVIEW generated file, and reading from the respective lines, as seen in figure.3.36. There needs to be the same amount of string variables as there are robot targets ($data1 \ldots data_n$). Since the RAPID-code is restricted to 80 characters.

```
PROC WriteData()
    Open "HOME:Hei2.txt", File\Read;
    data:= ReadStr(File);
    data2 := ReadStr(File\Line:=2);
    data3:= ReadStr(File\Line:=3);
    data4 := ReadStr(File\Line:=4);
    data5 := ReadStr(File\Line:=5);
    data6 := ReadStr(File\Line:=6);
    data7 := ReadStr(File\Line:=7);
    data8:= ReadStr(File\Line:=8);
    data9 := ReadStr(File\Line:=9);
    data10:= ReadStr(File\Line:=10);
    data11 := ReadStr(File\Line:=11);
    data12 := ReadStr(File\Line:=12);
    data13:= ReadStr(File\Line:=13);
    data14 := ReadStr(File\Line:=14);

    Close File;
ENDPROC
```

Figure 3.36: Data Allocation

Further steps needs to be executed to declare the information into its respective robot targets. Here, a function is used due to the program's reason to run as many times as there are robot targets. To divide the information to the robot targets respective locations, the following steps are needed.

1. Declare all the variables needed to store the individual target data seen in figure 3.38.

```
VAR robtarget tmpTarget;
VAR bool bResults;
VAR num posX;
VAR num posY;
VAR num posZ;
VAR num posQ1;
VAR num posQ2;
VAR num posQ3;
VAR num posQ4;
VAR num poscf1;
VAR num poscf4;
VAR num poscf6;
VAR num poscfx;
```

Figure 3.37: Variable Declaration

2. The data is further located using the previous mentioned ";" separator and stored in its respective location. The code uses the knowledge of the string length, and previous length data string length to locate the correct data in the string as seen in figure 3.38.

```
! Find split possition in string
posX := StrFind(value,1,";");
posY := strFind(value,posX+1,";");
posZ := StrFind(value,posY+1,";");
posQ1 := StrFind(value,posZ+1,";");
posQ2 := StrFind(value,posQ1+1,";");
posQ3 := StrFind(value,posQ2+1,";");
posQ4 := StrFind(value,posQ3+1,";");
poscf1 := StrFind(value,posQ4+1,";");
poscf4 := StrFind(value,poscf1+1,";");
poscf6 := StrFind(value,poscf4+1,";");
poscfx := StrFind(value,poscf6+1,";");
```

Figure 3.38: Position Allocation

3. In the final step of the function, the data is allocated to its respective positions, as seen in figure.3.39.

```
!Pos data
bResults:=StrToVal(StrPart(value,1,posX-1),tmpTarget.trans.x);
bResults:=StrToVal(StrPart(value,posX+1,posY-posX-1),tmpTarget.trans.y);
bResults:=StrToVal(StrPart(value,posY+1,posZ-posY-1),tmpTarget.trans.z);


bResults:=StrToVal(StrPart(value,posZ+1,posQ1-posZ-1),tmpTarget.rot.q1);
bResults:=StrToVal(StrPart(value,posQ1+1,posQ2-posQ1-1),tmpTarget.rot.q2);
bResults:=StrToVal(StrPart(value,posQ2+1,posQ3-posQ2-1),tmpTarget.rot.q3);
bResults:=StrToVal(StrPart(value,posQ3+1,posQ4-posQ3-1),tmpTarget.rot.q4);

bResults:=StrToVal(StrPart(value,posQ4+1,poscf1-posQ4-1),tmpTarget.robconf.cf1);
bResults:=StrToVal(StrPart(value,poscf1+1,poscf4-poscf1-1),tmpTarget.robconf.cf4);
bResults:=StrToVal(StrPart(value,poscf4+1,poscf6-poscf4-1),tmpTarget.robconf.cf6);
bResults:=StrToVal(StrPart(value,poscf6+1,poscfx-poscf6-1),tmpTarget.robconf.cfx);

RETURN tmpTarget;
```

Figure 3.39: Target Generation

The main function of the RAPID-code is run as seen in figure 3.40. This code is used to utilize the above-mentioned function, to further use the defined robot targets to follow the desired path.

```
PROC main()
    WriteData;
     Target_init:= StringToTarget(data);
     Target_10 := StringToTarget(data2);
     Target_11 := StringToTarget(data3);
     Target_12 := StringToTarget(data4);
     Target_13 := StringToTarget(data5);
     Target_14 := StringToTarget(data6);
     Target_15 := StringToTarget(data7);
     Target_init2:= StringToTarget(data8);
     Target_20 := StringToTarget(data9);
     Target_21 := StringToTarget(data10);
     Target_22 := StringToTarget(data11);
     Target_23 := StringToTarget(data12);
     Target_24 := StringToTarget(data13);
     Target_25 := StringToTarget(data14);
    Path_20;
ENDPROC
```

Figure 3.40: Target Allocation

With the targets defined, the path can be executed, and the desired features can be milled. When using a robot to mill, it is highly important that the *zone* is set to $z0$ of *fine*. The *zone* zero command forces the path to go directly tough the target, whereas the *fine* command stops in the target for a brief moment. The path used, can be seen in figure.3.41.

```
PROC Path_20()
    MoveJ Target_Home,v500,z100,Mill_UiA\WObj:=wobj0;
    MoveL Target_init,v500,z0,Mill_UiA\WObj:=wobj0;
    MoveL Target_11,v200,z0,Mill_UiA\WObj:=wobj0;
    MoveC Target_10,Target_12,v20,z0,Mill_UiA\WObj:=wobj0;
    MoveL Target_14,v20,z0,Mill_UiA\WObj:=wobj0;
    MoveC Target_15,Target_13,v20,z0,Mill_UiA\WObj:=wobj0;
    MoveL Target_11,v20,z0,Mill_UiA\WObj:=wobj0;
    MoveL Target_init,v200,z0,Mill_UiA\WObj:=wobj0;
    MoveL Target_init2,v200,z0,Mill_UiA\WObj:=wobj0;
    MoveL Target_21,v200,z0,Mill_UiA\WObj:=wobj0;
    MoveC Target_20,Target_22,v20,z0,Mill_UiA\WObj:=wobj0;
    MoveL Target_24,v20,z0,Mill_UiA\WObj:=wobj0;
    MoveC Target_25,Target_23,v20,z0,Mill_UiA\WObj:=wobj0;
    MoveL Target_21,v20,z0,Mill_UiA\WObj:=wobj0;
    MoveL Target_init2,v500,z0,Mill_UiA\WObj:=wobj0;
    MoveL Target_Home2,v500,z100,tool0\WObj:=wobj0;
ENDPROC
```

Figure 3.41: Path Execution

Using the approach described above; a fully automated robot trajectory can be generated to a satisfactory degree, and the method can be further developed to account for lock-box, lock, handles and more hinges.

3.9 Milling

When working with wooden doors which are to be installed in modern buildings, an exact and continuous surface-finish of high quality is desirable. Therefore, two different milling approaches were evaluated:

- Rough pass, with high speed, followed by a finishing pass to ensure good surface finish.
- One pass, with lower feed, to ensure good surface finish.

In the experiments; it was decided to use the latter, due to the material removal rate was low. Thus, low feed rate, and climb-milling is desirable. The reason climb-milling is the best solution is because of its tendency to transfer heat into the chips, and not to the work-piece. Further, the chips can evacuate behind the tool, which allows for a refined surface finish. The drawback with climb-milling is its tendency to drag the mill forward into the work piece, resulting in backlash[1] of the gearbox gears, which again can cause fluctuations. However, in the robot configuration chosen throughout this project, the weight of the machine is grater then the forces transferred from the mill, so this effect can be neglected.

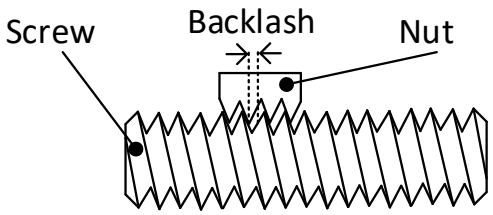
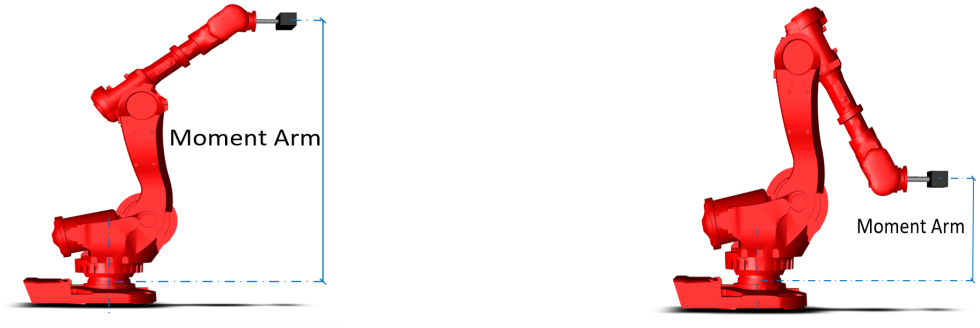


Figure 3.42: Backlash

When working with a robot in a milling application, achieving the same rigidity as in a CNC machine is not realistic. In order to avoid large sudden forces, due to the spindle inertia, the milling process has to be adapted. If this is neglected, tool fluctuations and uneven surfaces may be the results. One effective way to adapt the process is to, if possible, move the work object close to the robot. This reduces the arm of the load, which can greatly reduce the generated torque on the robot joints. This effect is illustrated with the figures 3.43a and 3.43b.



(a) Excessive Torque (b) Reduced Torque

Figure 3.43: Generated Torque Visualization

[1]Back-lash is the phenomenon which occurs when a screw or a gear are changes direction. The motion will then be lost between the screw and nut or two gears in the direction change, until the gears are again mated. This play is critical to ensure sufficient gear clearance. Thus, anti-back-lash nuts and gears are often used in modern CNC Machines

It was decided to investigate an empirical approach, with calculated initial data. It is critical that the Surface Feet per Minute (SFM)[2] value is within its given range. (The SFM value for soft wood is desired at 700 - 900 SFM [17]). This is achieved by a balance between the spindle speed and feed-rate. The milling tool chosen for this application was rated to speeds up to 18 000 RPM. However, an speed of 18 000 RPM requires a high feed-rate to maintain the SFM value, which again causes inertia-fluctuations. Therefore, it was decided to reduce the feed-rate to a point where the milling direction changes did not cause spindle fluctuations. By running a "Dry-Test" with the desired path, but no milling, the effect of the spindle inertia caused fluctuations. The feed could be reduced to a level where the oscillations were no longer noticeable. This experiment yielded a desired federate of $\approx 70mm/s$ and was the target feed for the experiment.

The milling tool used was a 19mm two-flute straight wood-mill. The data-sheet for this particular tool was not available, but data for similar mills states the chip load should be in the $0.019" - 0.025"$ range. Data sheet is found in appendix A.23.

The following metric conversion was made, as seen in table.3.4. The calculations were done to determine a starting-point for the spindle velocity and feed of the milling tool [18].

Table 3.4: Metric Conversion

	SFM[$\frac{ft}{min}$]	SMM[$\frac{m}{min}$]	CL[In]	FPT[mm]
Surface Speed	600 - 800	182 - 305	–	–
Chipload	–	–	0.019-0.025	0.4826 - 0.635

$$RPM = \frac{1000 \cdot SMM}{\pi \cdot d} \tag{3.51}$$

$$MMPM = FPT \cdot T \cdot RPM \tag{3.52}$$

where:

SMM	-	Surface Meters per Minute
d	-	Mill Diameter[mm]
$MMPM$	-	Feed [$\frac{mm}{min}$]
FPT	-	Feed per Tooth [mm]
T	-	Number of Teeth in Cutter

[2] *SFM - Also called surface speed or simply speed is the speed difference (relative velocity) between the cutting tool and the surface of the work piece it is operating on. It is expressed in units of distance along the work piece surface per unit of time, typically surface feet per minute (sfm) or meters per minute*(m/min). [17]

4 | Results

In this chapter the achieved results are presented. The results from the stiffness analysis, together with comparison to the simulation model are presented, followed by the results from evaluation of pre loading influence, link stiffness and kinematic performance. The resulting region of operation is presented and finally the results of the path generation and milling are presented.

Robot Stiffness

The results from the joint stiffness identification experiments are presented in table 4.1. The results are normalized to the highest stiffness, which are found in joint 1. In these experiments a static load was used, and position measurements with the Faro laser tracker, were logged in ten seconds before the load was applied and then for ten seconds after applying the load, when the system had reached steady state. The results are calculated using mean values from both the ATI Omega force sensor, and the Faro laser tracker.

Table 4.1: Estimated Joint Stiffness Normalized relative to Joint 1

	Joint 1	**Joint 2**	**Joint 3**	**Joint 4**	**Joint 5**	**Joint 6**
Joint Stiffness Normalized	1	0.887	0.6892	0.141	0.048	0.0187

Stiffness Simulation

Displacements logged with the Faro laser tracker during the experiments were compared to displacements in a simulation model developed in SimulationX. The identified joint stiffness was implemented in the simulation model, together with the respective load conditions. The displacement results from the simulation and comparison to displacements from the stiffness identification are shown in table 4.2.

Table 4.2: Comparing Simulated and Measured Displacement

	Joint 1	**Joint 2**	**Joint 3**	**Joint 4**	**Joint 5**	**Joint 6**
Displacement Measured [μm]	65.7	107.5	223.5	88.1	708.0	541.7
Displacement Simulated [μm]	68.9	107.9	145.9	109.0	561.8	473.2
Deviation [μm]	0.5	0.6	77.6	20.9	146.2	68.5
Deviation [%]	0.76	0.56	34.72	23.72	20.65	12.65

Pre-loading of Joints

An experiment was conducted to check for difference in the stiffness of a joint if it is pre-loaded compared non-loaded. To make the joint either pre-loaded or non-loaded, the joint was jogged into position from both directions. When the joint is jogged from the side that the load would be acting, the joint is considered pre-loaded. When jogged in the same direction that the load would be acting, the joint is considered non-loaded. The results of this experiment are shown in table 4.3. The results in table 4.3, are also normalized to the highest stiffness in the complete stiffness analysis.

Table 4.3: Stiffness Joint 1 Pre-loaded and Non-loaded

	Pre Load 1	**Pre Load 2**	**Not Pre Load 1**	**Not Pre Load 2**
Joint Stiffness Normalized	1.015	1.125	0.758	0.713

Influence of Complementary Stiffness Matrix

The ratios ν_p and ν_r are calculated for all values within the range of joint 2 and 3, to investigate if the complementary stiffness matrix ($\boldsymbol{K}_c$) has any influence on the Cartesian stiffness matrix ($\boldsymbol{K}_x$). This is done to evaluate if $\boldsymbol{K}_c$ can be neglected when evaluating the robot stiffness. The contour plots in figure 4.1 show that there are robot configurations where $\boldsymbol{K}_c$ will have some influence on $\boldsymbol{K}_x$, both in position (left plot), and in rotation (right plot). However, the values are very small; $\nu_p \leq 9.42 \cdot 10^{-4}$ m and $\nu_r \leq 1.85 \cdot 10^{-4}$ °.

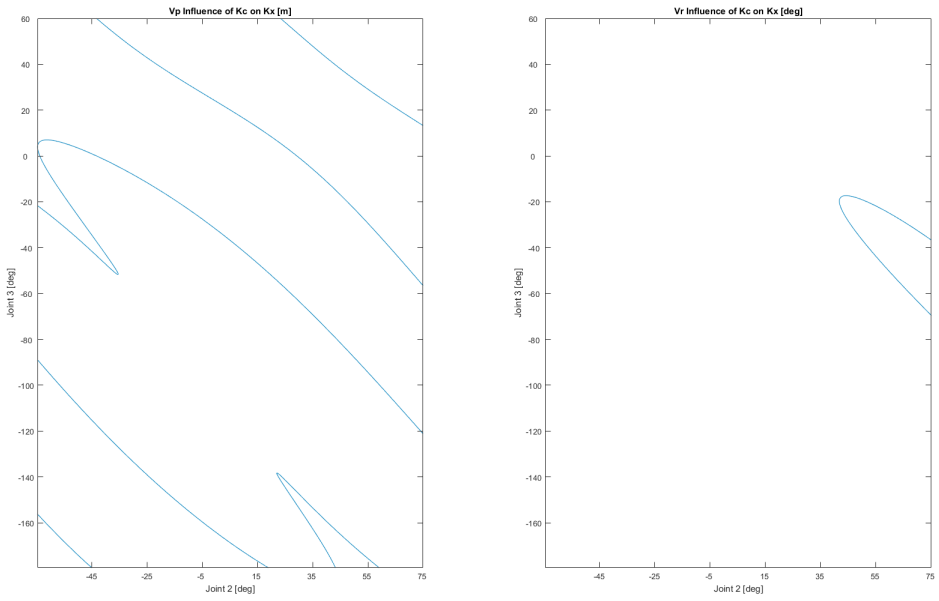


Figure 4.1: Influence of $\boldsymbol{K}_c$ on $\boldsymbol{K}_x$ in Position and Rotation

Kinematic Performance

The contour plot in figure 4.2 show the inverse Frobenius norm based condition number of the normalized Jacobian matrix $\boldsymbol{J}_n$, in the joint space of joint 2 and 3. High numbers (green contours) indicates a configuration where the robot has good dexterity, and low numbers (blue contours) indicates a configuration where the robot is closer to singularities. It is therefore desired to fit the robot region of operation to where its configuration falls within the green contours of the plot.

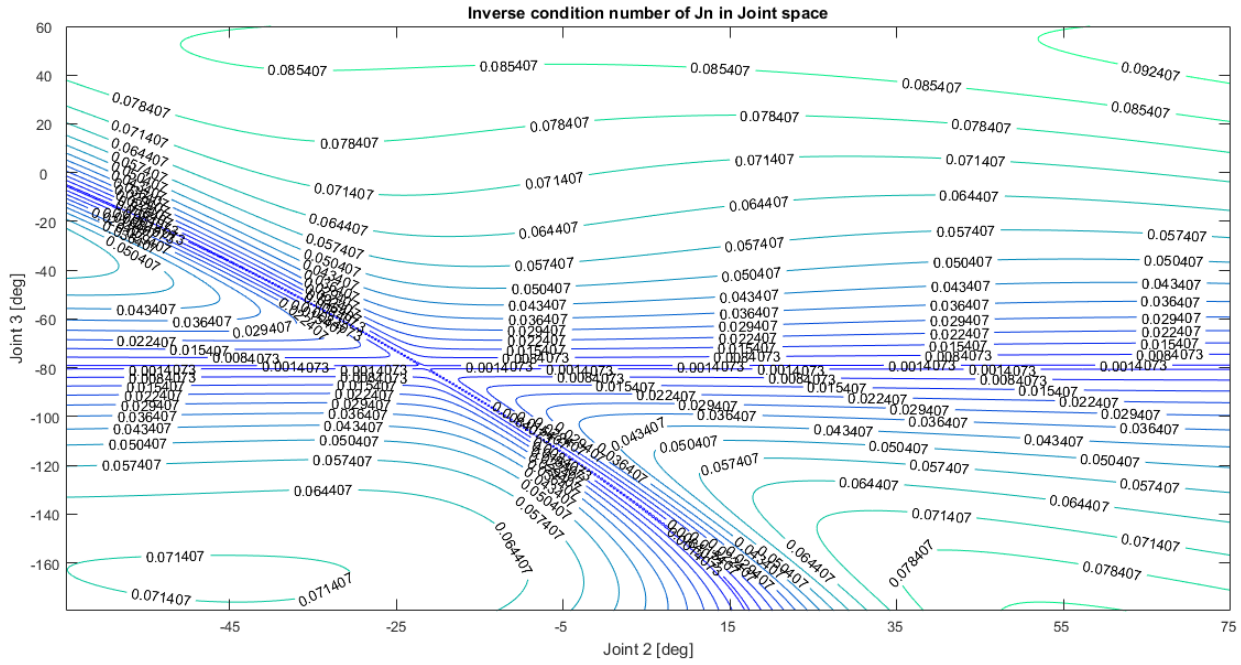


Figure 4.2: Contour Plot of Inverse Condition Number of $\boldsymbol{J}_n$

Region of Operation

Using the equations described in method section 3.6, the work space of the robot is given by required translational movement in x-,y- and z-direction shown in table 4.4:

Table 4.4: Required Translational Movement to Reach all Door Sizes

	δx $[mm]$	δy $[mm]$	δz $[mm]$
Translational Movement	100	580	2700

Using the inverse kinematics derived in theory section 2.1.2, the robot joint angles are found, based on desired end effector pose and robot configuration. The joint angles are then checked to be within valid range for the robot, and also if joint 2 and 3 are in a good area compared to the contour plot in figure 4.2. This was done iteratively, and the results shown in table 4.5, presents the region of operation for the robot related to the robot coordinate system.

Table 4.5: Region of Operation

Along	x-axis $[mm]$	y-axis $[mm]$	z-axis $[mm]$
Position Range Related to the Robot	1300 - 1400	-290 - 290	0 - 2700

Table 4.6 show the joint angles in degrees when the robot is in position for top and bottom hinge, for both the widest and narrowest door.

Table 4.6: Joint Angles

	Joint 1	**Joint 2**	**Joint 3**	**Joint 4**	**Joint 5**	**Joint 6**
Bottom Hinge $W_{d,max}$	-13.58	69.54	34.97	-14.01	-104.10	-93.48
Bottom Hinge $W_{d,min}$	13.58	69.54	34.97	14.01	-104.10	-86.52
Top Hinge $W_{d,max}$	-13.58	6.89	-44.31	21.68	39.46	-107.06
Top Hinge $W_{d,min}$	13.58	6.89	-44.31	-21.68	39.46	-72.94

The Robot has to be mounted on a pedestal, and the height of this pedestal is determined of the height of the clamping device for the door.

Path Generation

The sensor method chosen was the axis vision set-up presented in section 3.7. Here, only once camera was used to extract the 3-DOF. The camera method in combination with the target generation system presented in section 3.8 yield satisfactory results, as seen in table 4.7. Numerous configurations within the door localization limits ($\pm 10mm$ and $\pm 2°$) were tested without any errors. The different configurations are shown in figures 4.3a. - 4.3c.

Table 4.7: Path Generation Results

	Execution Time $[ms]$	**Accuracy** $[mm]$
System Results	800	0.2

(a) Configuration 1 (b) Configuration 2 (c) Configuration 3

Figure 4.3: Various Configuration for the Door

Robot Trajectory

The results of the path generation sequence were tested and verified using Robot Studio. The four following paths were generated as shown in figure 4.4.

- Door data: $Height = 2500mm$, $Width = 190mm$ and $Thickness = 50mm$
  Origin offset $X = Z = 0$ $Yaw = 0°$ (White)
- Door data: $Height = 2500mm$, $Width = 190mm$ and $Thickness = 50mm$
  Origin offset $X = Z = 0$ $Yaw = 2°$ (Blue)
- Door data: $Height = 2500mm$, $Width = 1350mm$ and $Thickness = 50mm$
  Origin offset $X = Z = 0$ $Yaw = 0°$ (Red)
- Door data: $Height = 2500mm$, $Width = 1350mm$ and $Thickness = 50mm$
  Origin offset $X = Z = 0$ $Yaw = 2°$ (Green)

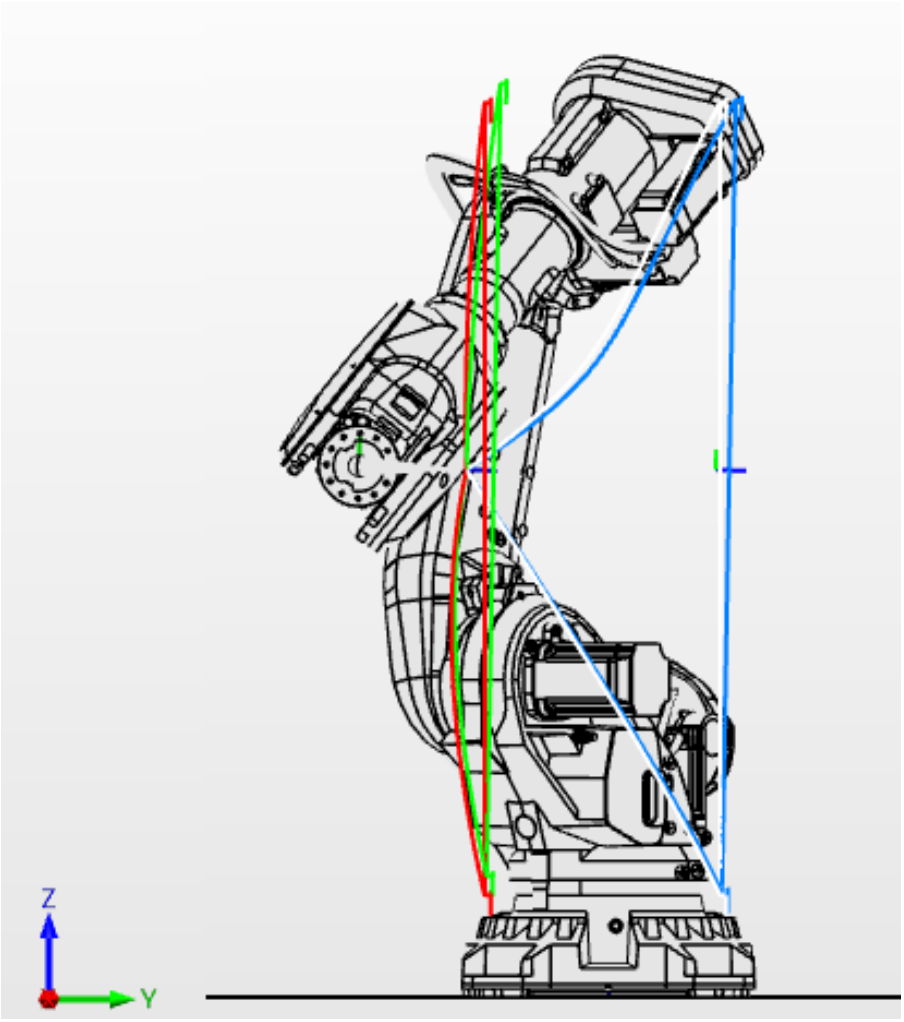
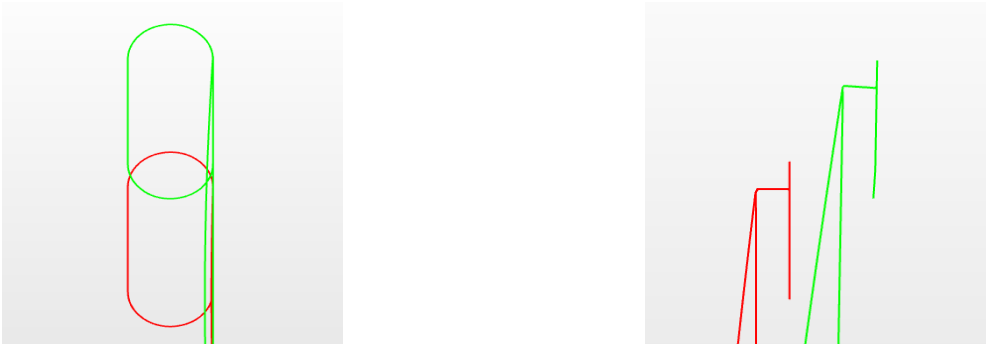


Figure 4.4: Generated Robot Paths

It can be seen, that the height difference between the fillets from the widest door (red and green lines) is larger than the smallest (white and blue lines). This acts an identification that

the utilized equations work as attended. The hinge path for the two bottom hinges can be seen in ZX-plane and ZY-plane in figure 4.5a and figure 4.5b, respectively. Here it can be seen that the desired geometry mentioned in section **??**. is achieved.



(a) Hinge Path ZX-Plane

(b) Hinge Path ZY-Plane

Figure 4.5: Hinge Path

TCP Deflection

Due to relatively soft material (wood) the deflection of the tool-tip was not excessive during milling. In figure 4.6 forces acting on one of the fillet lines can be seen, here the weight of the spindle itself is included. The data contains deflection (from the top) in X,Y and Z coordinates[m], as well as Roll, Pitch and Yaw in radians, these forces can be seen in table 4.8, and mean values is found using the MATLAB®script in appendix A.16.

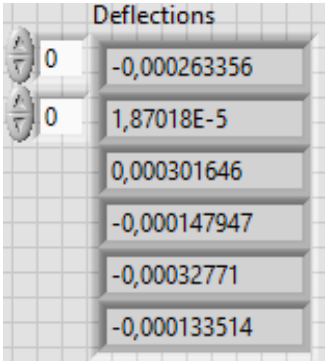


Figure 4.6: Milling Deflections

Table 4.8: Extracted Milling Forces

	F_X[N]	F_Y[N]	F_Z[N]	M_X[Nm]	M_Y[Nm]	M_Z[Nm]
Left Line	−209	29	12	−5	−43	3
Top Turn	−100	13	25	−3	−26	2
Right Line	−62	3	−11	−1	−8	2
Bottom Turn	−13	2	−22	−1	4	0

Milling

The initial and final feed-rate together with the spindle velocity are shown in table 4.9. The results of the initial starting-point yielded unsatisfactory surface-finish and marks of fluctuation, as seen in figure.4.7a. This was probably caused by the spindle inertia, seeing the work-piece did not fluctuate. The feed-rate was lowered to 50mm/s and the RPM to 3200. With these values; a satisfactory surface finish was achieved, as shown in figure.4.7b.

Table 4.9: Feed-Rate and RPM

	Initial	Final
Spindle Speed $[RPM]$	5025	3200
Feed-Rate $[\frac{m}{s}]$	80	50

(a) Uneven Surface-Finish

(b) Even Surface-Finish

Figure 4.7: Compare Surface-Finish

5 | Discussion

In processes where the execution time is of high importance, a factor connecting time and cost is always present. Throughout this report, the camera has chosen as the sensor method. However, if the price of photoelectric sensor or the Zivid camera could be defended, they would yield faster pose data of the door.

A joint stiffness analysis was conducted on the ABB IRB 6600. The results are not given directly in this report, as they are considered to be sensitive information. Hence, the presented results are normalized relative to joint 1. The stiffness analysis for joint 1 includes the base, motor with controller, and gearbox. The stiffness identified in joint 2 includes the link between joint 2 and 3, motor with controller, and gearbox. In addition, to better isolate joint 2, the load was attached directly to joint 3. The deviation between the measured and simulated joint displacement is as little as $0.5\mu m$ and $0.6\mu m$, for joint 1 and 2, respectively. This indicates that the stiffness identification methods are valid.

The analysis of joint 3 includes the links between joint 3 and 5, motor with controller, and gearbox. However, the applied load will also cause a deflection in joint 2, and the displacement due to this deflection is included in the estimation of joint 3. The deviation between the measured and simulated joint displacement is larger than for the 2 first joints, but $77\mu m$ is still considered to be a result which indicates a valid stiffness identification. One reason for the deviation could be that the applied load may cause a small deflection in joint 5, which is not included in the estimation.

To analyse the stiffness in joint 4 and 5, the local angular deflection in the respective joint is measured. For both of these analyses, the applied load will cause a deflection in joint 2 and 3, which influence the total measure displacement. This is compensated for by calculating the angle for the evaluated joint before and after applying the load. The deviation between the measured and simulated displacement in the analysis is $20.9\mu m$ and $146.2\mu m$, for joint 4 and 5 respectively. Joint 5 have the largest deviation between the measured and simulated displacement. However, this is also the analysis which has the largest displacements measured. In addition, joint 2 and 3, which are significantly stiffer than joint 5, are influencing the result in this analysis. Nevertheless, the deviations are not very large, which indicates that the identification methods are valid also for these joints.

When the stiffness in joint 6 was evaluated, an external arm was mounted in order to create a moment in the joint. The load applied to this arm will also cause a displacement in joint 1, which will influence the total measured displacement. This is included in the evaluation of joint 6. The deviation between the measured and simulated displacement is $68.5\mu m$, which indicates that also this stiffness identification method is valid.

The conducted experiments regarding pre-loading of the joints, show that there is a vast difference between pre-loaded compared to non-loaded joints, when estimating the joint stiffness. Therefore, in the conducted joint stiffness identification, all 6 joint were intentionally pre-loaded after releasing the brakes. The results can therefore be used for all joint angles, as long as the

joint can be considered loaded. The only time a joint is not considered loaded, is when it changes direction of rotation.

The influence of $\boldsymbol{K}_c$ on $\boldsymbol{K}_x$ is also evaluated. The result shows that there are regions where $\boldsymbol{K}_c$ has some influence, but the influence is very small. The wrench used to evaluate the influence also contains forces substantially larger than the forces logged during milling. Therefore it was assumed that the complementary stiffness matrix could be neglected for this project.

An evaluation of the kinematic performance based on the Jacobian number was also performed. The result from this evaluation is a contour plot which shows regions for joint 2 and 3 where the robot is far from singularities. This was considered when choosing the region of operation for the robot. It was also ensured that the robot could reach all possible hinge positions for the different door sizes. This region is presented, and if an ABB IRB 6600 should be used for this application, it has to be mounted on a pedestal. The height of the pedestal depends on the height of the door clamping device.

The generated code for designing the milling path worked as intendant. The vision system data was verified by using a calliper to establish the offset deflection. Further, the entire process of generating a complete path in RAPID code, by using initial door parameters worked successfully. Due to the simple geometry of the features, a direct path generation is coded in RAPID. If complex geometry is desired, a g-code converter to RAPID code should be implemented. The automatic communication between the respective programs can be further developed, but the approach used between NI LabVIEW and RAPID to test the concept was successful.

Throughout this thesis, the door deviation has been fixed to 3 DOF. However, if more freedom of the work-piece is needed, all the presented sensor methods can be further developed to account for the pose of the door.

One point which could be improved is the extraction of force data from the milling process, used for off-line compensation. Our approach yielded results quite obstructed by noise. An alternative approach would be to insert the tool of choice, with the desired feeds and speeds, in a rigid mill with a force sensor installed. These results would be less prone to noise, due to higher stiffness, and the task of locating the forces relative to the hinge-points would be attained with ease.

However, both on-line compensation and closed-loop sensor compensation would be feasible solutions for a milling application, the implementation of a closed-loop controller is not possible using the ABB-interface. In addition, in our application, the price/time/result aspect is not predicted to yield better results than the inexpensive off-line compensation. The two above mentioned methods are highly interesting in the presence of complex geometry and large variety in milling paths. If the computational time used to generate the milling path proves too extensive in comparison to the placement time of the door, a method where this data is computed beforehand can be explored.

As the robot used in this thesis is a robot used for a variety of different projects, and the last project was not a milling application. Therefore the spindle had to be mounted on the robot, and wired. In addition, the tool cabinet had to be rewired.

The process force path correction was implemented at the end of the project. Hence, this has not been tested. However, proper equipment used to measure the required accuracy of $\pm 0.5mm$

was not available. Further, the largest compensation deviation as seen in figure 4.6, in the result section is $0.3mm$, which indicates that the path correction yields feasible correction data.

Regarding milling, further attention must be directed to work-piece clamping, tooling and path optimisation. There is little need spending time and effort in milling path compensation if the work-piece clamping is not rigid or the tooling is not optimal, e.g. too small, large, wrong geometry etc. When working with soft materials i.e. wood, plastics and foam, stiffness compensation often yields neglectable values. However, if large rate of material removal is desired, the generated forces can result in deflections worth accounting for. Also, if the robot is in a configuration where the inertia generated by the spindle is high, due to high feed-rates, the deflection of the robot will be greater and the deflection compensation must be accounted for.

6 | Conclusion

In this master thesis, a joint stiffness analysis was conducted on the ABB IRB 6600 robot located in the mechatronics laboratory at UiA. The results from this analysis were verified using a simulation model in SimulationX. Comprising analysis and simulation results indicates that the used stiffness identification methods are valid.

Evaluating the influence of complementary stiffness matrix on the Cartesian stiffness matrix showed that the complementary stiffness matrix could be neglected for this robot with the measured process forces, and the estimated stiffness for each joint was used to correct the generated path trajectory.

A space of operation where the robot has good dexterity, is far from singularities, and is able to reach all door sizes, is found. This is achieved by evaluating the Jacobian number, range of robot joints and knowing the variety of door sizes. To achieve the best possible end-product, configuration, path, milling velocity, and feed rate, are all considered.

The entire process regarding the automatic generation of the robot target includes several parts which must communicate to generate both an effective and executable path trajectory for the robot. The camera system which is responsible for determining the placement deviation has been tested, and yielded results well within the required accuracy of 0.5 mm. The execution time of the process is less than 800 ms, which is concluded to be fast enough to noticeably influence the cycle time.

The calculations used to account for both X, Z and Yaw-displacement were derived with success, and the robot spindle pitch is also derived in quaternions. The path correction based on the robot stiffness is also added in this step. A ready-to-read RAPID file is further generated in NI LabVIEW, and is posted in a common folder. This *$.txt$ file contains the path and information regarding the milling actions, which the robot use to mill the desired feature in the door, based on provided door data.

Bibliography

[1] A. A. T. AB, *Product specification Articulated Robot IRB 6600*, ABB, Jan. 2018. [Online]. Available: https://library.e.abb.com/public/560fa420555c2d8ac1257b4b0052112c/3HAC023933-001_rev1_en.pdf

[2] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, C. F. Schultz, Ed. John Wiley & Sons, inc., 2006.

[3] D. L. Pieper, "The kinematics of manipulators under computer control," Stanford University, Tech. Rep., 1968.

[4] Omron, "Photoelectric sensors," https://www.ia.omron.com/support/guide/43/introduction.html, May 2018.

[5] S. Roland, N. I. R, and S. Davide, *Autonomous Mobile Robots*, 2nd ed. MIT Press, 2011, isbn: 978-0-262-01535-6.

[6] I. Tyapin, G. Hovland, and T. Brogårdh, "Method for estimating combined controller, joint and link stiffnesses of an industrial robot," 2014.

[7] S. Chen, T. Zhang, and M. Shao, "A 6-dof articulated robot stiffness research," 2016.

[8] E. Abele, M. Weigold, and S. Rothenbücher, "Modeling and identification of an industrial robot for machining applications," 2007.

[9] C. Dumas, S. Caro, MehdiCherif, S. Garnier, and B. Furet, "Joint stiffness identification of industrial serial robots," 2011.

[10] S.-F. Chen, "The 6x6 stiffness formulation and transfort-nation of serial manipulators via the cct theory," 2003.

[11] U. Schneider, M. Momeni-K, M. Ansaloni, and A. Verl, "Stiffness modeling of industrial robots for deformation compensation in machining," pp. 4464–4469, Sept 2014.

[12] F. O. Rasch, *Sponskjærende bearbeidng - 5.* Universitetsforlaget, 1973.

[13] KennaMetal, "Work hardening during machining," https://www.kennametal.com, May 2018.

[14] S. Zargarbashi, W. Khan, and J. Angeles, "The jacobian condition number as a dexterity index in 6r machining robots," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 6, pp. 694 – 699, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0736584512000427

[15] G. G. H. and V. Loan, *Matrix Computations, 3rd ed*, J. Hopkins, Ed. Baltimore, 1996.

[16] A. Khan, Waseem, and J. Angeles, "The kinetostatic optimization of robotic manipulators: The inverse and the direct problems," vol. 128, 01 2006.

[17] P. Heritage, *FRESING metoder og arbeidsteknikk*, T. Mårds, Ed. Tiden Norsk Forlag, 1986.

[18] H. Hartvigsen, R. Lorentsen, K. Michelsen, and S. Seljevoll, *Verksted Håndboka*. Gyldendal Undervisning, 2014.

A | Appendix

A.1 System Description

ROBOTNORGE

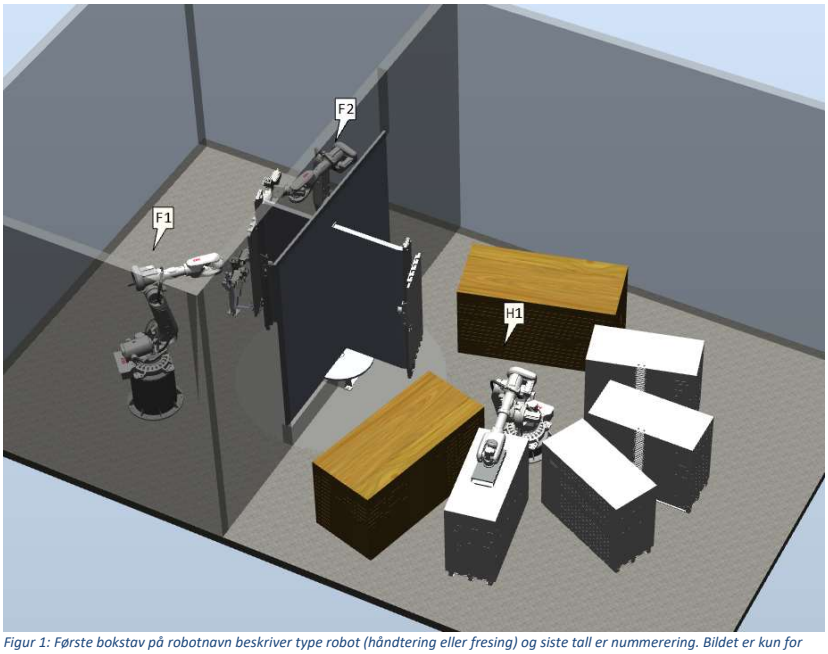
Nordic Dørfabrikk
Kvavik
4580 LYNGDAL

Deres ref.: Jan Steinar Reiersen

Skrevet av: Gaute Nærland Serigstad/Ivan Rott
Dato: Mandag 29 Januar 2018
Revisjon: Revisjon 05 – En-celle anlegg m/svingdør

Systembeskrivelse - Robotanlegg til utsparing av Dører

Robotanlegget er et system som bearbeider lommer og hull til hengsler, håndtak og låser i ulike tredører, med varierte material sammensettinger.



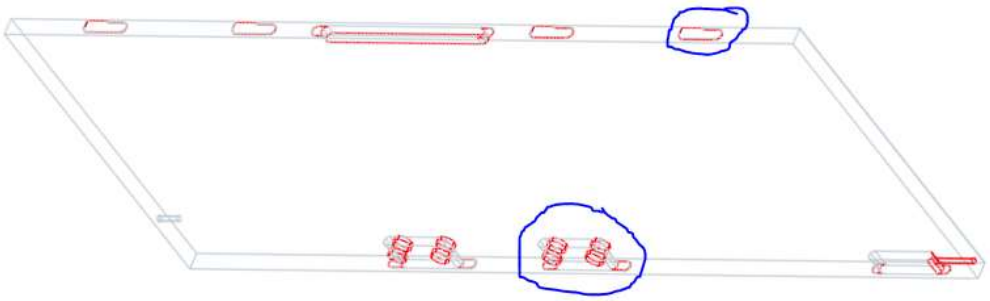
Figur 1: Første bokstav på robotnavn beskriver type robot (håndtering eller fresing) og siste tall er nummerering. Bildet er kun for illustrasjon.

Innhold

1. Prosjektorganisasjon

Nordic Door AS: Jan Steinar Reiersen
RobotNorge AS: Gaute N. Serigstad / Kjetil Sævareid

2. Generell beskrivelse av anlegget



Figur 2: Utsparinger som inngår i standard arbeidsstykke (modul lås og hengsel)

Vi har tatt utgangspunkt i standard arbeidsstykke og avgrenset omfanget til håndtering, oppmåling og utfresing av 1 stk. variant nedfellbar hengsl i 3 stk. posisjoner og 1 stk. modul lås i midten på dør. Bearbeidingstider brukt i syklustid er hentet fra video filmet på befaring med overnevnt utgangspunkt.

Håndteringsrobot: H1
Freseroboter: F1 og F2

Operatøren i anlegget setter x-antall paller med dørblader inn i robotcellen. Når pallene er satt i posisjon, og operatør har gått ut av robotcelle, kan prosessen i robotcellen startes. Håndteringsrobot identifiserer, måler og håndterer dørbladene med å lese strekkoden på det øverste dørbladet. Den løfter det øverste dørblad inn i svingdøren, deretter roterer svingdøren 180 grader og dørbladet er klar for bearbeiding.

Freseroboter både presisjonsmåler og bearbeider dørbladet. Samtidig repeterer håndteringsrobot sin prosess med neste produkt fra sin plukkestasjon. Etter endt bearbeiding roterer svingdøren tilbake 180 grader og håndteringsrobot kan løfte et ferdig bearbeidet dørblad ut fra svingdør og ned til leveringsposisjon. Ved repeterbar prosess stables dørbladene oppå hverandre til definert max-antall for stabel. Ved bruk av svingdøren er det mulig for håndteringsrobot og freseroboter å arbeide samtidig.

Når enten leveingsposisjon er full, eller plukkeposisjon er tom, må en operatør inn og bytte de nødvendige pallene før robot kan fortsette prosessen. I tillegg er det tilrettelagt for palle for dører som skal videre til glass, eller dører som er feilvarer. En kontainer til å riste støv i er også tilrettelagt for.

Videre vil det være mulighet for å bygge på med baner inn og ut av robotcelle, ved bruk av baner er det også muligheter for enda en robotcelle på motsatt side av håndteringsrobot.

Variasjon i arbeidsstykkene:

Minimum dimensjoner på håndterbare arbeidsstykker:
Høyde-1000mm x Bredde-290mm x Tykkelse-30mm
Maks dimensjoner på håndterbare arbeidsstykker:
Høyde-3000mm x Bredde-1350mm x Tykkelse-100mm
(*Hentet fra «Teknisk spesifikasjon ny CNC» dokumentet tilsendt fra kunde, 13. Juli 2017)*

Maks totalvekt på produkter er 125kg.

Ved dører med høyde over 1200mm er det nødvendig med L-verktøy for bearbeiding av skåter i topp og bunn. Vanlig bearbeiding langs sidene på dør klarer begge robotene med full høyde (3000mm).

3. Simulering/Estimering av syklustider forventet til anlegget

Utgangspunkt for våre simuleringer og estimater på syklustider er tatt utfra fra info/data mottatt og verifisert mot «Standard dør» (Uten utsparing på topp/bunn) som følgende :

Frese prosess:
Vi har tatt utgangspunkt i filmer tatt i produksjon.

Utfresing av hengsler bruker 33 sekunder.
Utfresing av lås/dørhåndtak 15 sek per hull = 60 sek
Utfresing av låsekasse 24 sek

Dersom vi fordeler de forskjellige jobbene på 2 roboter blir det slik:
Robot 1 freser 2 hull til lås/dørhåndtak fra den ene siden + hengsel (syklustid 63 sek)
Robot 2 freser 2 hull til lås/dørhåndtak + låsekasse (syklustid 54 sek)

Det vil si vi har 17 sekunder igjen for å klare kravet på 80 sek. Måling før bearbeiding og rotasjon på svingdør vil spise av de resterende sekundene.

Søking:
Ved å bruke analoge sensorer til søking vil vi kunne søke opp døren på under 10 sek.
63 (freseprosess) + 10 (oppmåling) = 73 sek

Rotasjon på svingdør:
Dermed gjenstår 7 sekunder til å kunne rotere svingdøren. Den aktuelle hastigheten til svingdør er enda ikke avklart.

4. Nordic Door AS sine forpliktelser

RobotNorge vil innen rimelig tid og i samarbeid med kunden definere kundens forpliktelser og leveranser forbundet med gjennomføring av avtalt testomfang under FAT, installasjon og IAT. Eventuelle reisekostnader og transportkostnader forbundet med dette er ikke en del av leveransen.

Dersom ikke annet er særskilt avtalt skal alt forberedende arbeid på produksjonssted hos kunde være utført i.h.t. tegninger og prosjektplan før leverandørens montører ankommer.

Kunden er ansvarlig for at det er tilstrekkelig styrke i gulv og andre fundamenter før anlegget skal monteres for igangkjøring.

Kunden skal blant annet sørge for:

- At montasjestedet er klargjort til avtalt tidspunkt.
- At anleggets komponenter er ført frem til det klargjorte montasjested.
- At nødvendig løfteutstyr er tilgjengelig.
- At personell er tilgjengelig for praktisk bistand, herunder for eksempel: truckkjøring, boring og bolting i eget fabrikkgulv, kjøring av annet utstyr, enkle forandringer i grensesnitt.
- At sikkerhetssystem og signal grensesnitt til eksisterende utstyr er på plass.
- At infrastruktur som kabelbaner/ kabelføringsveier, signalkabler osv. er ført frem.
- At strømtilførsel er ført frem.
- At tilførsel for pneumatikk er ført frem.
- At nødvendig tilførsel av luft for ventilering og avsug er installert eller påtenkt.
- Kabelbaner/ kabelføringsveier mellom el. skap og robotcelle.
- Sikre nødvendig opplæring av drift og vedlikeholds personell.

5. Robotprogram og programmering av flere varianter

Håndteringsrobot søker opp posisjonen til den øverste døren på stabel og leser strekkode. Robotstyringen vil deretter hente ut resept fra egen database. Dimensjoner og posisjoner for utfresinger. Resept databasen lages på et egenprodusert PC program. Størrelser på dørblad, varianter av hengsler og låser med posisjon på dørblad kan hentes fra eksisterende ERP, CAM Software eller Excel Dok.

Freseroboter søker opp lokalt koordinatsystem på dør, samt nøyaktig posisjon for hengsler og dørlås. Deretter freses dørlås, dørhåndtak og låskasse samtidig som hengsler freses på motsatt side.

Personell hos RobotNorge bruker RobotStudio offline programmering i prosjekt og igangkjøringsfasen. Hele anlegget bygges opp, simuleres og testes i forkant. Alle bevegelser med signaler, logikk blir programmert og testet. Når anlegget er installert og funksjonstestet, lastes de ferdige programmene inn i robotsystemet.

RobotNorge vil forberede programmet til å kunne ta inn parameter til å justere bane/utsparring basert på kjente felles-parametere for hengslene og dørlåser (Nordic Door må gi input for gjeldende parametere).

Nordic Door må ta frem nye resepter ved programmering av flere varianter produkt og nye produkt når de kommer og blir aktuelle.

Vi har utført en enkel simulering i RobotStudio på standard arbeidsstykke. Simuleringen viser at prosessen med handtering og utfresing av 1 stk. variant nedfellbar hengsl, i 3 stk. posisjoner og modul lås på standard arbeidsstykke, tar i underkant av 80 sek. Ved å ha 2 identiske celler vil dette gi en syklustid på ca. 40 sek pr. dør.

Svingdør blir styrt av robot.

6. Leverandørens krav til miljø og installasjonssted

Anlegget er ment å arbeide under følgende forhold:
Temperaturområde: 5-45 grader Celsius.
Luftfuktighet: RF<95 %.

7. Leveranseomfang

7.1 Roboter

Håndteringsrobot: 1 stk. IRB 6700 – 175/3.05, 6-akse robotmanipulator

- Stander utførelse, lakkert og med kapslingsgrad innenfor krav på IP65.
- Grovoppmåling og håndtering
- Strekkode-lesing

Freseroboter: 2 stk. IRB 6700 – 235/2.65, 6-akse robotmanipulator

- Stander utførelse, lakkert og med kapslingsgrad innenfor krav på IP65.
- Presisjonsmåling og bearbeiding m/blåsing

7.2 Sokler til robotmanipulatorer

Alle sokler er i stål og lakkert sort, og de må festes til gulv med kjemiske ankerbolter.

- 1 stk. IRB 6700. Riser 300mm til håndteringsrobot
- 1 stk. IRB 6700. Riser 1000mm til freserobot 1
- 1 stk. IRB 6700. Riser 1100mm til freserobot 2

7.3 IRC5 styresystem til hver robot

- Ethernet
- 1 stk. programmeringsenhet (FlexPendant) med 10 meter kabel.
- 3 stk. opptil 30 meter lange kabler mellom robotmanipulatorer og styreskaper.
- Kjølevifter bak på robotskap har kapslingsgrad IP33. Robotskapet må installeres og brukes i et normalt industrimiljø innendørs, hvor omgivelsestemperaturen ligger i området fra +5 °C til + 45 °C og luftfuktigheten <95%, ikke kondenserende.

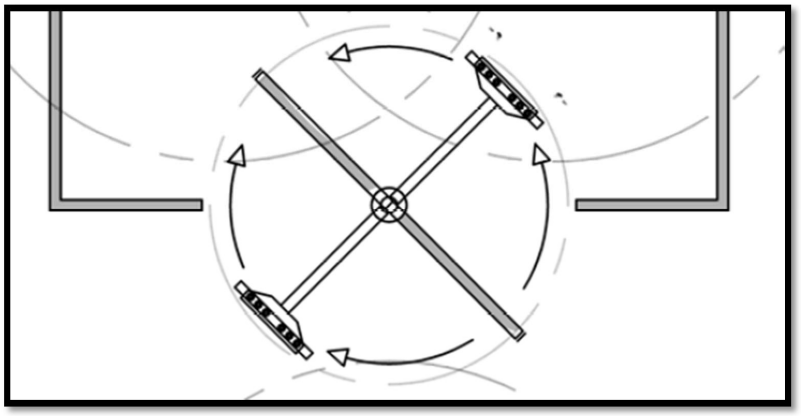
Figur 3: Styreskap

Figur 4: FlexPendant

7.4 Svingdør

Håndteringsrobot leverer dørbladene i svingdøren, som roterer 180 grader inn til freseroboter. Svingdøren består av 2 stk. gripere som gjør det mulig å forberede neste fresejobb samtidig som freseroboter bearbeider på motsatt side.

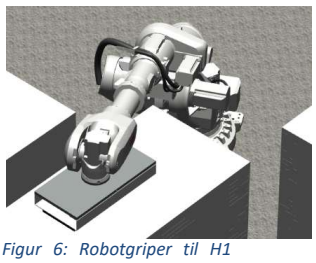
- Robotcellen leveres med 1 stk. svingdør
- Svingdøren støtter 2 stk. dørblad om gangen, en på utsiden og en på innsiden
- Svingdøren blir montert på 1 stk. MID 1000 som roterer svingdøren 180 grader begge veier
- 2 stk. Avanserte og like gripere, en på utsiden og en på innsiden



Figur 5. Illustrasjonsbilde av svingdør

7.5 Verktøyer

Robotgriper til IRB 6700 – 175/3.05 (Håndteringsrobot)
Verktøyet har leser for strekkode, og leser hvert unike dørblad før bearbeiding. I tillegg er verktøyet utstyrt med en sensor som roboten bruker til å måle dørbladets posisjon med, før den griper dørbladet i senter og løfter dett inn i støttejigg. Etter bearbeiding løfter verktøyet dørbladene ut av robotcelle og videre til programmert leveringsposisjon.

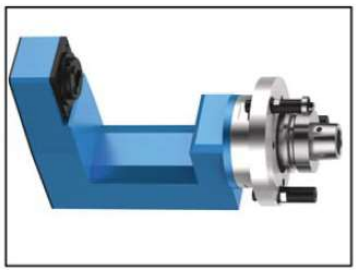


Figur 6: Robotgriper til H1 (illustrasjon)

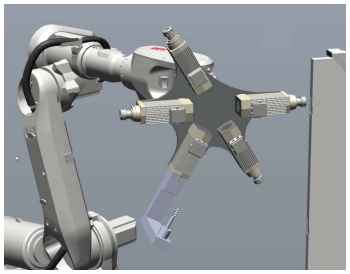
Robotverktøy IRB 6700 – 235/2.65 (Freseroboter)
Verktøyet har to sensorer som roboten bruker til å presisjonsmåle dørene med, før bearbeiding. Robotens program er ut fra teoretiske referanser og den kompenserer for avvik på døren i både bredde, høyde og diagonalt.

For fjerning av spon har verktøyet blåsedyse montert ved spindelen. Fjerner spon fra verktøy og bearbeidingsområde på dør.

Verktøyet har **4 stk**. motorspindler KRSV 51,14-2D med spennhylser for manuelt bytte av verktøy. 5,0kW, 300Hz, 17500rpm. Disse skal frese hull til håndtak og lomme til hengsler. I tillegg er det **1 stk.** «L-verktøy». Eventuelt 1 stk. ekstra motorspindel i stedet for «L-verktøyet».

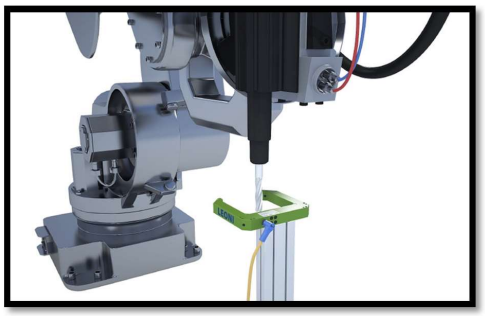


Figur 7: Illustrasjonsbilde av L-verktøy



Figur 8: Illustrasjonsbilde av robotgriper

1 stk. Kalibreringssystem (Advintec TCP)
Advintec TCP-verktøyets kalibreringssystem kalibrerer vertøyer eller fixturer. Robotprogrammet korrigeres automatisk av de målte variasjonene og sikrer at verktøyet alltid arbeider i riktig posisjon. Advintec TCP er et høyt presisjonskalibreringssystem med en nøyaktighet på 0,02mm.



Figur 9: Advintec TCP

Freseroboter blir utstyrt med felles kalibreringssystem. Hver gang det har vært verktøybytte vil de kjøre innom kalibreringssystemet og kalibrere verktøyet selv. Operatør må informere robot om at verktøy er byttet, eller ved andre anledninger hvor det er behov for kalibrering av verktøy.

7.6 Elektrisk spesifikasjon

Anlegget leveres med/ for følgende elektriske spesifikasjon:

Tilførselspenning Robot:	400VAC, 50Hz, 32 A*
Tilførselsspenning Fres:	
Tilførselspunkt(er):	Hovedbryter i styreskap

*) Vi anbefaler jordfeilbryter av klasse **A-si** eller klasse **B** der dette er påkrevd.

7.7 Overflatebehandling

Styreskap:	Galvaniserte plater, Grå front.
Roboter:	Standard grå.
Deler i aluminium:	Ubehandlet.
Deler i stål:	Sort lakk.

7.8 Sikkerhet

RobotNorge AS har ansvar for nødstopp brytere på programmeringsenheten og på robotstyreskap.

RobotNorge AS har ikke tatt med noen løsning av sikkerhet i form av hus eller gjerde rundt robotcelle.

7.9 El. Skap

1 stk. stander stålskap: H1800, B800 og D300. Komplett med:

- Hovedbryter 4-polt 25A
- 10 stk. frekvensomformere 9kW til 5kW motorspindler (230V 1-fas forsyning)
- Styresignal mot omformere tilkobles IO node (start/stopp og feil)
- Nødstoppkontaktor i forkant av omformere
- Nødstopprele til Safe Lock
- IO node for robotens ethernet/IP.

- Kabel for signalutveksling
- Kabler til motorspindler m/innfester
- Styring til svingdør

7.10 Dokumentasjon

Dokumentasjonen leveres senest 4 uker etter overtagelse av anlegget. Dokumentasjonen leveres elektronisk på CD, minnebrikke eller som nedlastbar link.

7.11 Demonstrering/Testing av tilbudt simulert prosess-løsning

Før oppstart av prosjektet vil RobotNorge, etter avtale, kjøre en kvalifisering/verifikasjon av tilbudt simulert prosess-løsning ved hjelp av en test hos Orkanger (Ref. oversendt tilbud). De har både robot (IRB 6700 – 235/2.65) og spindel tilgjengelig.

Hensikten med testen blir å validere identifiserte risikoelementer:

- Roboten som stabil arbeidsplatform for freseverktøyet
- Robotens evne til å måle opp arbeidsstykket nøyaktig
- Prosesskreftene under bearbeiding
- Stabilitet til dør «fastspent i jigg» med unigrippere (vackuumputer)
- Oppmåling av arbeidsstykket med robot
- Syklustid og kvalitet
- Eventuelt forsøke å identifisere prosess grensen

Det forventes at Nordic Door sender testpaneler/dører til Orkanger, etter avtale.
Vi vil også måtte lage en «testprosedyre» som forteller hva og hvordan vi skal teste og som skal godkjennes av Nordic Door.

Betingelser:
Alt.1 Dersom testen blir vellykket og Nordic Door bestiller anlegget, skal Nordic Door ikke betale for testen.
Alt.2 Dersom testen blir vellykket men Nordic Door bestiller en annen løsning vil vi fakturer kostnaden for testen (350 KNOK)
Alt.3 Dersom testen ikke skulle bli vellykket skal Nordic Door ikke betale for testen

7.12 Prosjektledelse

- Utredning og godkjenning av endelig leveranseomfang og layout.
- Kvalitet sikring av løsning med 3d CAD modell av komplett robotanlegg inkludert robot, el. skap og pallplasser i SolidWorks og test med program i RobotStudio.
- Generell prosjektstyring med ansvar for prosjektplan med milepæler.
- Koordinering og planlegging av nødvendige ressurser.
- Oppfølging under montasje og innkjøring.
- Koordinering av funksjonstester.

- Utarbeidelse av dokumentasjonsmateriale.
- I leveransen inngår ett oppstartmøte med Nordic Door AS hos RobotNorge.

7.13 Testkjøring på Klepp/ FAT

Robot med robotverktøy monteres på Klepp for funksjonstest/ FAT før anlegget demonteres og klargjøres for forsendelse. Testen er planlagt over en dag og skal overvåkes og godkjennes av personell fra NORDIC DOOR AS.

Nordic Door er forpliktet til å sende dører, pinnefreser og hylser til testkjøring/FAT.

Testomfang:

- X-antall plukke og leveringsposisjoner
- Komplett svingdør og roboter oppstilt ihht layout
- Funksjonstest av robotverktøy
- Testkjøring av 3 dører med 3 ulike produkt varianter av «Styrofoam».
- Lese strekkode for produktvariant og verifisere at robot kjører riktig program
- Det utstedes test dokument som godkjennes av Nordic Door AS før pakking og forsendelse

7.14 Montering hos Nordic Door AS

Vi har planlagt 2 personer i to uker til installasjon og funksjonstest og 1 stk. person i tre uke til oppstart og trimming av komplett anlegg i automatikk, med 1 variant produkt.

Personell fra RobotNorge monterer og kobler opp anlegget på produksjonsstedet. Personell fra RobotNorge ønsker adgang og praktisk mulighet til å utføre arbeid i minst 10 timer/ døgn under installasjonsperioden, alle dager unntatt lørdag og søndag.

7.15 Igangkjøringstest hos kunde/ IAT

Etter oppstart og trimming av komplett anlegg i automatikk, med produkt gjennomføres en overtagelsestest (IAT) av anlegget.

- Testen er tilsvarende FAT, men med alle lokale forhold inkludert, plukkestasjoner, «hus» til freseroboter, sikkerhet, lys, betjening av hele anlegget osv. Testen utføres med to personer i løpet av en dag, med 1 variant av produkt i produksjon.

Overtagelsespapirer signeres etter gjennomført og godkjent test.
Ut fra erfaring vil det kunne være behov mindre justeringer og fintrimmingen av anlegget i tiden etter at robotcella er overtatt og satt i produksjon. I denne perioden kan det være behov for support fra service personell hos RobotNorge. Fjernoppkopling kan være en enkel løsning.

7.16 Operatøropplæring på anlegget etter overlevering

Det blir enkel operatør opplæring.
Opplæringen vil i hovedsak bestå av gjennomgang av sikkerhetsbeskrivelse og brukerhåndbok, samt praktiske øvelser på anlegget. (10-15 timer)

Robot kurs bestilles separat ved behov.

- Operatør kurs RAPID/ IRC5
- Grunnkurs programmering IRC5 / RAPID
- Videregående programmering IRC5
- Kurs i RobotStudio

7.17 Samsvar og CE merking

RobotNorge leverer samsvarserklæring for delvis ferdigstilte maskiner for sin leveranse. Denne eller disse benyttes i den endelige CE dokumentasjonen.

8. Ikke evaluert/priset i tilbudet

- Håndtering av spon ut fra robotcelle
- Børsting av dørblader etter bearbeiding, før lakkering
- Hus til freseroboter
- Sikkerhet rundt robotcelle

9. Risikoelementer identifisert

- Toleranser og finish på prosess
- Vibrasjoner i robotverktøy og/eller svingdør ved utfresing

10. Nordic Door AS vil blant annet stå ansvarlig for:

- Overordnet prosjektledelse
- Alle nødvendige input-data for prosjekteringen (Størrelser, varianter, etc)
- Vedlikeholdsrutiner og data på fres-varianter
- Freser av de ulike varianter for produkter-spekteret som skal kjøres gjennom stasjonen
- Hylse mellom Pinnefreser (ND) og Spindel (RN)
- Håndtering av støv og spon. Sørge for at dører kommer frie for støv og spon ut av frese-hus, samt det som måtte forekomme på innsiden av frese-hus

11. Leveringsplan

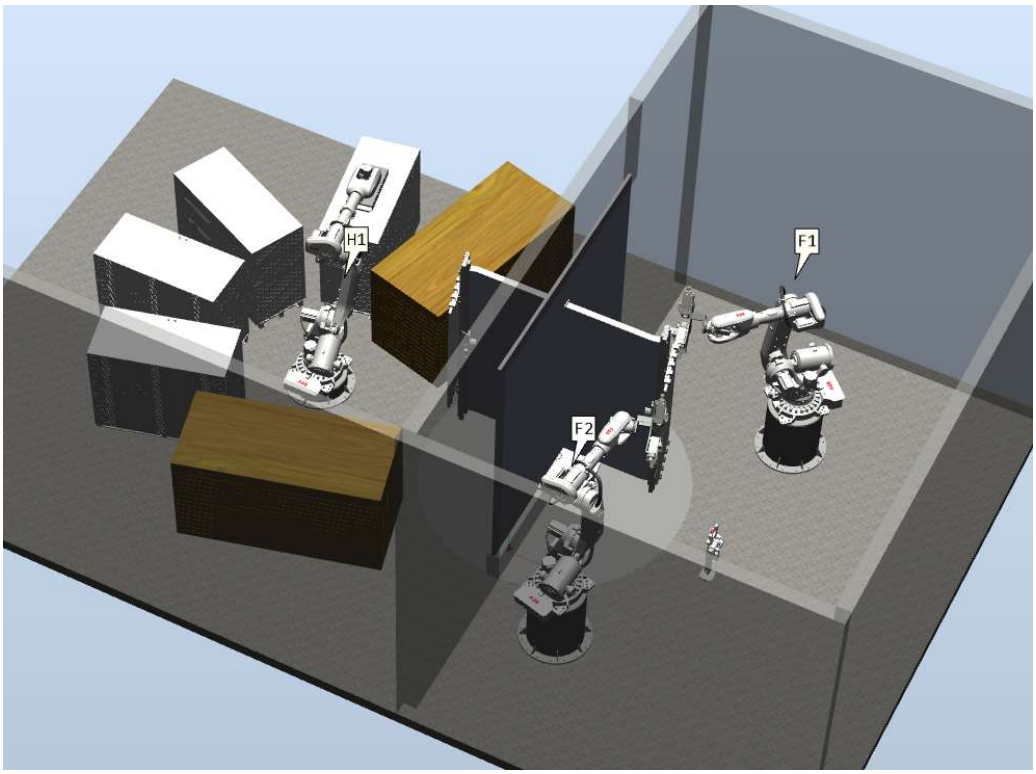
Plan for gjennomføring av prosjektet vil typisk ha følgende hovedmilepæler/-aktiviteter:

• Signert ordrebekreftelse / kontrakt (Ordredato)	…….
• Layout og teknisk spesifikasjon godkjent	1 - 2 arbeidsuker etter ordredato.
• Design Approval og prosjektplan godkjent	6 - 8 arbeidsuker etter ordredato.
• Roboter fra Fabrikk levert RobotNorge	ca. 15-17 uker fra ordre
• Støttejigg-komponenter mottatt fra leverandør	ca. 16-18 uker fra ordre
• Funksjonstester hos RobotNorge på Klepp	ca.27-29 uker fra ordre
• Frakt av utstyr til NORDIC DOOR AS	Etter nærmere avtale
• Start mekanisk og elektrisk montasje	
• Start funksjonstest	

- Start trimming av komplett anlegg i automatikk
- Produksjonsstart
- Overtakelse
- Support fra service personell hos RobotNorge i oppstartsperioden.

Detaljert leveringsplan og fremdriftsplan følger protokoll fra Design Approval.

12. Layout



Figur 10: Første bokstav på robotnavn beskriver type robot (håndtering eller fresing) og siste tall er nummerering. Bildet er kun for illustrasjon.

A.2 Forward Kinematics

```
% Forward Kinematics ABB IRB6600-2.55

clear all;
close all;
clc;

% Link Lengths [m]

L1 = 0.780;
L2 = 0.320;
L3 = 1.075;
L4 = 0.200;
L5 = 1.142;
L6 = 0.200;

% Joint Angles [rad]
% When all angles are 0, the robot is in home position
Q = [0 0 0 0 0 0];

q1  = Q(1);
q2  = Q(2);
q3  = Q(3);
q4  = Q(4);
q5  = Q(5);
q6  = Q(6);

% Denavit-Hartenberg table for the IRB 6600

A1=RotZ(q1)*TransZ(L1)*TransX(L2)*RotX(-pi/2);
A2=RotZ(q2-pi/2)*TransZ(0)*TransX(L3)*RotX(0);
A3=RotZ(q3)*TransZ(0)*TransX(L4)*RotX(-pi/2);
A4=RotZ(q4)*TransZ(L5)*TransX(0)*RotX(pi/2);
A5=RotZ(q5)*TransZ(0)*TransX(0)*RotX(-pi/2);
A6=RotZ(q6+pi)*TransZ(L6)*TransX(0)*RotX(0);

% The transformation matrix
T = A1*A2*A3*A4*A5*A6
```

```
T =

   -0.0000   -0.0000    1.0000    1.6620
   -0.0000    1.0000    0.0000    0.0000
   -1.0000   -0.0000   -0.0000    2.0550
         0         0         0    1.0000
```

Published with MATLAB® R2017a

```
function T=RotZ(q)
cq=cos(q);
sq=sin(q);
T = [cq -sq 0 0 ; sq cq 0 0 ; 0 0 1 0 ; 0 0 0 1];

function T=RotX(q)
cq=cos(q);
sq=sin(q);
T = [1 0 0 0 ; 0 cq -sq 0 ; 0 sq cq 0 ; 0 0 0 1];

function T=TransX(L)
T = [1 0 0 L ; 0 1 0 0 ; 0 0 1 0 ; 0 0 0 1];

function T=TransZ(L)
T = [1 0 0 0 ; 0 1 0 0 ; 0 0 1 L ; 0 0 0 1];
```

A.3 Inverse Kinematics

```
% Link Lengths
L1=0.78;
L2=0.320;
L3=1.075;
L4=0.200;
L5=1.142;
L6=0.200;
LtoolX=0;
LtoolZ=0;

% Define the robot configuration
conf = [0 0 0];
% conf(1)=0 means base upwards, conf(1)=1 means base backwards
% conf(2)=0 means elbow up, conf(1)=1 means elbow down
% conf(3)=0 means wrist up, conf(1)=1 means wrist down

% Define the position and arientation for joint 6
H = [0 0 1 1.662;0 1 0 0;-1 0 0 2.500;0 0 0 1];

Tuframe = eye(4);
T = inv(Tuframe)*H;

X=T(1:3,4);                 %X,Y,Z position of tool in user frame
P=Tuframe*[X;1];            %X,Y,Z position of tool in global frame

%Tool. Define tool if desired.
Lx=0;
Ly=0;
Lz=0;
Rt=eye(3);                  %Tool mouting orientation

Ru=Tuframe(1:3,1:3);        %Orientation of user-frame
Rl=T(1:3,1:3);              %User-specified tool orientation

Ro=Ru*Rl*Rt';
V=Ro*[Lx;Ly;L6+Lz];          %Vector from WP to TCP

% WP locations
X5 = real(P(1) - V(1));
Y5 = real(P(2) - V(2));
Z5 = real(P(3) - V(3));

Q=zeros(1,6);
Q(1)=atan2(Y5,X5);          % Add base backwards if nessesary
Reachable='Yes';            % Set to false if out of reach

% Optimized code from Maple
% code for wrist position
t1 = cos(Q(1));
t2 = t1 ^ 2;
t3 = sin(Q(1));
t4 = t3 ^ 2;
t6 = 0.1e1 / (t2 + t4);
xc1 = t1 * t6 * X5 + t3 * t6 * Y5 - L2;
t11 = pi / 0.2e1;
t12 = cos(t11);
t14 = t12 ^ 2;
t16 = sin(t11);
t17 = t16 ^ 2;
```

```
t22 = 0.1e1 / (t2 * t14 + t14 * t4 + t2 * t17 + t17 * t4);
t23 = t22 * X5;
t26 = t22 * Y5;
t29 = 0.1e1 / (t14 + t17);
yc1 = L1 * t16 * t29 - t16 * t29 * Z5 + t1 * t12 * t26 - t3 * t12 * t23;
zc1 = -L1 * t12 * t29 + t12 * t29 * Z5 + t1 * t16 * t26 - t3 * t16 * t23;

% Solving for th3 and th2

FirstLink  = L3;
SecondLink = sqrt(L4^2+L5^2);
D  = (xc1^2+yc1^2+zc1^2-FirstLink^2-SecondLink^2)/(2*FirstLink*SecondLink);
th3Off = pi - (pi/2) - atan2(L4,L5);


if(conf(2)==0)
       alpha= atan2(sqrt(1-D^2),D);
       th3 = (alpha-th3Off);
       beta = atan2(SecondLink*sin(alpha),FirstLink+SecondLink*cos(alpha));
       thetaB = atan2(sqrt(xc1^2+zc1^2),-yc1);
       th2 = thetaB - beta;
          Q(2) = th2;
          Q(3) = th3;
    else
       alpha= atan2(-sqrt(1-D^2),D);
       th3 = (alpha-th3Off);
       beta = atan2(SecondLink*sin(alpha),FirstLink+SecondLink*cos(alpha));
       thetaB = atan2(sqrt(xc1^2+zc1^2),-yc1);
       th2 = thetaB - beta;
          Q(2) = th2;
          Q(3) = th3;
    end

    if -45*pi/180 < th2 < 90*pi/180
    Reachable='Yes';
    else
    Reachable='No';
    end

    if -60*pi/180 < th3 < 60*pi/180
    Reachable='Yes';
    else
    Reachable='No';
    end

% Inverse kinematics for wrist variables

    r11 = Ro(1,1);
    r12 = Ro(1,2);
    r13 = Ro(1,3);
    r21 = Ro(2,1);
    r22 = Ro(2,2);
    r23 = Ro(2,3);
    r31 = Ro(3,1);
    r32 = Ro(3,2);
    r33 = Ro(3,3);

% Optimized code for wrist position from Maple

t1 = pi / 0.2e1;
t2 = -Q(2) + t1;
```

```
t3 = cos(t2);
t4 = cos(Q(1));
t5 = t3 * t4;
t6 = cos(t1);
t7 = t6 ^ 2;
t8 = cos(Q(3));
t9 = t7 * t8;
t10 = t5 * t9;
t11 = sin(t1);
t12 = t11 ^ 2;
t13 = t8 * t12;
t15 = t4 * t7;
t16 = sin(Q(3));
t17 = sin(t2);
t18 = t16 * t17;
t19 = t15 * t18;
t20 = t4 * t17;
t21 = t16 * t12;
t23 = sin(Q(1));
t24 = t23 * t3;
t25 = t6 * t16;
t27 = t23 * t6;
t28 = t8 * t17;
t31 = t23 ^ 2;
t32 = t3 ^ 2;
t33 = t31 * t32;
t34 = t16 ^ 2;
t35 = t7 * t34;
t37 = t8 ^ 2;
t38 = t7 * t37;
t40 = t34 * t12;
t42 = t37 * t12;
t44 = t31 * t7;
t45 = t17 ^ 2;
t46 = t34 * t45;
t48 = t37 * t45;
t50 = t31 * t45;
t53 = t4 ^ 2;
t54 = t32 * t53;
t59 = t53 * t7;
t62 = t53 * t45;
t65 = t33 * t35 + t33 * t38 + t33 * t40 + t33 * t42 + t54 * t35 + t54...
    * t38 + t50 * t40 + t54 * t40 + t62 * t40 + t50 * t42 + t54 * t42...
    + t62 * t42 + t44 * t46 + t44 * t48 + t59 * t46 + t59 * t48;
t66 = 0.1e1 / t65;
t69 = t24 * t9;
t71 = t23 * t7;
t72 = t71 * t18;
t73 = t23 * t17;
t76 = t4 * t6;
t84 = t32 * t7;
t85 = t84 * t34;
t86 = t84 * t37;
t87 = t32 * t34;
t89 = t32 * t37;
t91 = t7 * t45;
t92 = t91 * t34;
t93 = t91 * t37;
t94 = t46 * t12;
t95 = t48 * t12;
r = -(-t5 * t13 - t20 * t21 + t24 * t25 - t27 * t28 - t10 - t19) * t66...
```

```
    * r13 + (t24 * t13 + t73 * t21 + t5 * t25 - t76 * t28 + t69 + t72)...
    * t66 * r23 - (t16 * t3 - t28) * t11 / (t87 * t12 + t89 * t12 + t85...
    + t86 + t92 + t93 + t94 + t95) * r33;
t100 = t23 * t32;
t103 = t23 * t45;
t106 = t7 * t6;
t107 = t106 * t16;
t109 = t25 * t12;
t113 = t28 * t12;
t116 = t7 ^ 2;
t117 = t116 * t34;
t119 = t116 * t37;
t121 = t35 * t12;
t124 = t38 * t12;
t127 = t12 ^ 2;
t128 = t34 * t127;
t130 = t37 * t127;
t132 = t31 * t116;
t141 = t33 * t117 + t33 * t119 + 0.2e1 * t33 * t121 + 0.2e1 * t33 *...
    t124 + t33 * t128 + t50 * t128 + t33 * t130 + t50 * t130 + t132...
    * t46 + t132 * t48 + 0.2e1 * t44 * t94 + 0.2e1 * t44 * t95;
t150 = t53 * t116;
t159 = t54 * t117 + t54 * t119 + 0.2e1 * t54 * t121 + 0.2e1 * t54...
    * t124 + t54 * t128 + t62 * t128 + t54 * t130 + t62 * t130 + t150...
    * t46 + t150 * t48 + 0.2e1 * t59 * t94 + 0.2e1 * t59 * t95;
t161 = 0.1e1 / (t141 + t159);
t164 = t32 * t4;
t167 = t4 * t45;
t197 = t46 * t116 + t48 * t116 + t117 * t32 + t119 * t32 + t128 * t32...
    + t128 * t45 + t130 * t32 + t130 * t45 + 0.2e1 * t40 * t84 + 0.2e1...
    * t40 * t91 + 0.2e1 * t42 * t84 + 0.2e1 * t42 * t91;
t198 = 0.1e1 / t197;
s = (t4 * t106 * t28 + t100 * t40 + t100 * t42 + t103 * t40 + t103...
    * t42 - t5 * t107 - t5 * t109 + t76 * t113 - t69 - t72) * t161...
    * r13 - (-t23 * t106 * t28 + t24 * t107 + t24 * t109 - t27 * t113...
    + t164 * t40 + t164 * t42 + t167 * t40 + t167 * t42 - t10 - t19)...
    * t161 * r23 - (t8 * t3 + t18 + t46 + t48 + t87 + t89) * t6 * t11...
    * t198 * r33;
t202 = t6 * t34;
t204 = t6 * t37;
t208 = t7 * t16;
t213 = t6 * t8;
t217 = t11 * (t100 * t202 + t100 * t204 - t20 * t13 - t15 * t28 + t27...
    * t18 + t5 * t208 + t5 * t21 + t24 * t213 + t27 * t46 + t27 * t48);
t231 = t11 * (-t73 * t13 - t164 * t202 - t164 * t204 - t76 * t18 + t24...
    * t208 + t24 * t21 - t5 * t213 - t71 * t28 - t76 * t46 - t76 * t48);
t239 = (-t12 * t17 * t16 - t12 * t3 * t8 + t85 + t86 + t92 + t93) * t198;
t = -t217 * t161 * r11 - t231 * t161 * r21 + t239 * r31;
u = -t217 * t161 * r12 - t231 * t161 * r22 + t239 * r32;
v = -t217 * t161 * r13 - t231 * t161 * r23 + t239 * r33;

TA = zeros(3,3);

TA(1,3) = r;
TA(2,3) = s;
TA(3,1) = t;
TA(3,2) = u;
TA(3,3) = v;

% Solving for th5, th4 and th6
```

```
        q6a=acos(TA(3,3));
        q6b=acos(TA(3,3))+pi/2;
        if(conf(3)==0) %Wrist up
                Q(4)=atan2(-TA(2,3),-TA(1,3));
                Q(5)=min(q6a,q6b);
                Q(6)=atan2(TA(3,2),-TA(3,1));
    else            %Wrist down
                Q(4)=atan2(TA(2,3),TA(1,3));
                Q(5)=-max(q6a,q6b)+(pi/2);
                Q(6)=atan2(-TA(3,2),TA(3,1));
    end

    if -90*pi/180 < Q(5) < 90*pi/180
    Reachable='Yes';
    else
    Reachable='No';
    end
```

Published with MATLAB® R2017a

A.4 Maple Optimization for Inverse Kinematics I

```
restart;
with(linalg);
```

```
[BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp, Wronskian, addcol,
    addrow, adj, adjoint, angle, augment, backsub, band, basis, bezout, blockmatrix, charmat,
    charpoly, cholesky, col, coldim, colspace, colspan, companion, concat, cond, copyinto,
    crossprod, curl, definite, delcols, delrows, det, diag, diverge, dotprod, eigenvals, eigenvalues,
    eigenvectors, eigenvects, entermatrix, equal, exponential, extend, ffgausselim, fibonacci,
    forwardsub, frobenius, gausselim, gaussjord, geneqns, genmatrix, grad, hadamard, hermite,
    hessian, hilbert, htranspose, ihermite, indexfunc, innerprod, intbasis, inverse, ismith,
    issimilar, iszero, jacobian, jordan, kernel, laplacian, leastsqrs, linsolve, matadd, matrix,
    minor, minpoly, mulcol, mulrow, multiply, norm, normalize, nullspace, orthog, permanent,
    pivot, potential, randmatrix, randvector, rank, ratform, row, rowdim, rowspace, rowspan,
    rref, scalarmul, singularvals, smith, stackmatrix, submatrix, subvector, sumbasis, swapcol,
    swaprow, sylvester, toeplitz, trace, transpose, vandermonde, vecpotent, vectdim, vector,
    wronskian]
```

(1)

```
with(CodeGeneration);
```

```
[C, CSharp, Fortran, IntermediateCode, Java, JavaScript, LanguageDefinition, Matlab, Names,
    Perl, Python, R, Save, Translate, VisualBasic]
```

(2)

```
RotZ(q) := matrix(4, 4, [[cos(q), -sin(q), 0, 0], [sin(q), cos(q), 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]) :
RotX(q) := matrix(4, 4, [[1, 0, 0, 0], [0, cos(q), -sin(q), 0], [0, sin(q), cos(q), 0], [0, 0, 0, 1]]) :
Trans(X, Y, Z) := matrix(4, 4, [[1, 0, 0, X], [0, 1, 0, Y], [0, 0, 1, Z], [0, 0, 0, 1]]) :
T1 := multiply(RotZ(q1), Trans(0, 0, L1), Trans(L2, 0, 0), RotX(-pi/2)) :
T2 := multiply(RotZ(q2 - pi/2), Trans(L3, 0, 0)) :
T3 := multiply(RotZ(q3), Trans(0, 0, L4), RotX(-pi/2)) :
T4 := multiply(RotZ(q4), Trans(L5, 0, 0))
```

$$\begin{bmatrix} \cos(q4) & -\sin(q4) & 0 & \cos(q4)\,L5 \\ \sin(q4) & \cos(q4) & 0 & \sin(q4)\,L5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(3)

```
> T04 := multiply(T1, T2, T3, T4) :
R04 := submatrix(T04, 1..3, 1..3) :
R04i := inverse(R04) :
R0 := matrix(3, 3, [[r11, r12, r13], [r21, r22, r23], [r31, r32, r33]]) :
TA := multiply(R04i, R0) :
cs := [r = TA[1, 3], s = TA[2, 3], t = TA[3, 1], u = TA[3, 2], v = TA[3, 3]] :
Matlab(cs, optimize)
```

```
t1 = sin(q1);
t2 = pi / 0.2e1;
t3 = -q2 + t2;
t4 = cos(t3);
t5 = t4 ^ 2;
t6 = t1 * t5;
t7 = cos(q3);
t8 = t7 ^ 2;
t9 = sin(q4);
t10 = t8 * t9;
```

```
t11 = sin(t2);
t12 = t11 ^ 2;
t13 = t10 * t12;
t15 = sin(q3);
t16 = t15 ^ 2;
t17 = t16 * t9;
t18 = t17 * t12;
t20 = sin(t3);
t21 = t20 ^ 2;
t22 = t1 * t21;
t25 = cos(q1);
t26 = t4 * t25;
t27 = cos(t2);
t28 = t27 ^ 2;
t29 = t28 ^ 2;
t30 = t29 * t7;
t31 = cos(q4);
t32 = t30 * t31;
t34 = t26 * t28;
t35 = t7 * t12;
t36 = t35 * t31;
t39 = t12 ^ 2;
t41 = t7 * t39 * t31;
t43 = t25 * t29;
t44 = t20 * t15;
t45 = t44 * t31;
t47 = t25 * t28;
t48 = t47 * t20;
t49 = t15 * t12;
t50 = t49 * t31;
t53 = t25 * t20;
t55 = t15 * t39 * t31;
t57 = t1 * t4;
t58 = t28 * t27;
t59 = t58 * t15;
t60 = t59 * t31;
t61 = t57 * t60;
t62 = t57 * t27;
t63 = t62 * t50;
t64 = t1 * t58;
t65 = t20 * t7;
t66 = t65 * t31;
t67 = t64 * t66;
t68 = t1 * t27;
t69 = t68 * t20;
t70 = t69 * t36;
t71 = t59 * t9;
t72 = t26 * t71;
t73 = t26 * t27;
t74 = t15 * t9;
t75 = t74 * t12;
t76 = t73 * t75;
t77 = t25 * t58;
t78 = t65 * t9;
t79 = t77 * t78;
t80 = t25 * t27;
t81 = t80 * t20;
```

```
t82 = t7 * t9;
t83 = t82 * t12;
t84 = t81 * t83;
t85 = t28 * t7;
t86 = t85 * t9;
t88 = t1 * t28;
t89 = t44 * t9;
t91 = t22 * t13 + t6 * t13 + t22 * t18 + t6 * t18 + t26 * t32 +
t26 * t41 + 0.2e1 * t34 * t36 + t43 * t45 + 0.2e1 * t48 * t50 +
t53 * t55 - t57 * t86 - t88 * t89 - t61 - t63 + t67 + t70 - t72 -
t76 + t79 + t84;
t92 = t1 ^ 2;
t93 = t92 * t5;
t94 = t29 * t16;
t95 = t31 ^ 2;
t96 = t94 * t95;
t98 = t9 ^ 2;
t99 = t8 * t98;
t100 = t99 * t39;
t102 = t8 * t39;
t103 = t102 * t95;
t105 = t16 * t98;
t106 = t105 * t39;
t108 = t16 * t39;
t109 = t108 * t95;
t111 = t92 * t29;
t112 = t21 * t8;
t113 = t112 * t98;
t115 = t112 * t95;
t117 = t21 * t16;
t118 = t117 * t98;
t120 = t117 * t95;
t122 = t92 * t21;
t126 = t122 * t100 + t93 * t100 + t122 * t103 + t93 * t103 + t122
* t106 + t93 * t106 + t93 * t109 + t111 * t113 + t111 * t115 +
t111 * t118 + t111 * t120 + t93 * t96;
t128 = t25 ^ 2;
t129 = t5 * t128;
t130 = t29 * t8;
t131 = t130 * t98;
t133 = t130 * t95;
t135 = t94 * t98;
t142 = t128 * t29;
t146 = t129 * t100 + t129 * t103 + t129 * t106 + t122 * t109 +
t129 * t109 + t142 * t113 + t142 * t115 + t142 * t118 + t129 *
t131 + t129 * t133 + t129 * t135 + t129 * t96;
t149 = t128 * t21;
t157 = t93 * t28;
t158 = t99 * t12;
t161 = t8 * t12;
t162 = t161 * t95;
t165 = t105 * t12;
t168 = t16 * t12;
t169 = t168 * t95;
t172 = t149 * t100 + t149 * t103 + t149 * t106 + t149 * t109 +
t142 * t120 + t93 * t131 + t93 * t133 + t93 * t135 + 0.2e1 * t157
* t158 + 0.2e1 * t157 * t162 + 0.2e1 * t157 * t165 + 0.2e1 * t157
```

```
* t169;
t174 = t92 * t28 * t21;
t179 = t129 * t28;
t185 = t128 * t28 * t21;
t190 = t174 * t158 + t179 * t158 + t185 * t158 + t174 * t162 +
t179 * t162 + t185 * t162 + t174 * t165 + t179 * t165 + t185 *
t165 + t174 * t169 + t179 * t169 + t185 * t169;
t194 = 0.1e1 / (t126 + t146 + t172 + 0.2e1 * t190);
t198 = t57 * t28;
t202 = t1 * t29;
t204 = t88 * t20;
t207 = t1 * t20;
t209 = t5 * t25;
t212 = t25 * t21;
t215 = t57 * t71;
t216 = t62 * t75;
t217 = t64 * t78;
t218 = t69 * t83;
t219 = t26 * t60;
t220 = t73 * t50;
t221 = t77 * t66;
t222 = t81 * t36;
t225 = -t209 * t13 - t212 * t13 - t209 * t18 - t212 * t18 + 0.2e1
* t198 * t36 + t202 * t45 + 0.2e1 * t204 * t50 + t207 * t55 + t26
* t86 + t57 * t32 + t57 * t41 + t47 * t89 - t215 - t216 + t217 +
t218 + t219 + t220 - t221 - t222;
t228 = t27 * t5;
t231 = t21 * t27;
t234 = t31 * t15;
t235 = t28 * t4;
t237 = t31 * t12;
t238 = t15 * t4;
t240 = t31 * t7;
t241 = t20 * t28;
t244 = t27 * t4;
t246 = t20 * t27;
t250 = t29 * t5;
t252 = t95 * t8;
t255 = t95 * t16;
t257 = t12 * t98;
t259 = t8 * t28 * t5;
t262 = t95 * t12;
t266 = t16 * t28 * t5;
t271 = t39 * t98;
t272 = t8 * t5;
t274 = t95 * t39;
t276 = t16 * t5;
t279 = t105 * t250 + t252 * t250 + t255 * t250 + t99 * t250 +
0.2e1 * t257 * t259 + 0.2e1 * t257 * t266 + 0.2e1 * t262 * t259 +
0.2e1 * t262 * t266 + t271 * t272 + t271 * t276 + t274 * t272 +
t274 * t276;
t280 = t21 * t29;
t285 = t112 * t28;
t290 = t117 * t28;
t299 = t105 * t280 + t271 * t112 + t274 * t112 + t271 * t117 +
t274 * t117 + t252 * t280 + t255 * t280 + 0.2e1 * t257 * t285 +
0.2e1 * t257 * t290 + 0.2e1 * t262 * t285 + 0.2e1 * t262 * t290 +
```

```
t99 * t280;
t302 = 0.1e1 / (t279 + t299) * r33;
r = t91 * t194 * r13 + t225 * t194 * r23 - (t10 * t228 + t10 *
t231 + t17 * t228 + t17 * t231 + t234 * t235 + t237 * t238 - t237
* t65 - t240 * t241 + t82 * t244 + t74 * t246) * t11 * t302;
t304 = t161 * t31;
t306 = t168 * t31;
t310 = t30 * t9;
t314 = t82 * t39;
t319 = t74 * t39;
t321 = t85 * t31;
t324 = t22 * t304 + t22 * t306 - t26 * t310 - t26 * t314 + t6 *
t304 + t6 * t306 - t53 * t319 - t57 * t321 - 0.2e1 * t34 * t83 -
t43 * t89 - t88 * t45 - 0.2e1 * t48 * t75 + t215 + t216 - t217 -
t218 - t219 - t220 + t221 + t222;
t341 = 0.2e1 * t198 * t83 + t202 * t89 + 0.2e1 * t204 * t75 +
t207 * t319 + t209 * t304 + t209 * t306 + t212 * t304 + t212 *
t306 - t26 * t321 + t57 * t310 + t57 * t314 - t47 * t45 + t61 +
t63 - t67 - t70 + t72 + t76 - t79 - t84;
t344 = t31 * t8;
t346 = t31 * t16;
t351 = t12 * t9;
s = t324 * t194 * r13 - t341 * t194 * r23 - (t344 * t228 + t346 *
t228 + t344 * t231 + t346 * t231 + t234 * t246 - t74 * t235 -
t351 * t238 + t240 * t244 + t82 * t241 + t351 * t65) * t11 *
t302;
t360 = t8 * t27;
t362 = t16 * t27;
t366 = t15 * t28;
t371 = t7 * t27;
t375 = (t112 * t68 + t117 * t68 + t366 * t26 + t49 * t26 - t35 *
t53 + t360 * t6 + t362 * t6 + t371 * t57 + t44 * t68 - t65 * t47)
* t11;
t392 = t102 * t122 + t102 * t93 + t108 * t122 + t108 * t93 + t112
* t111 + t117 * t111 + t130 * t93 + 0.2e1 * t161 * t157 + 0.2e1 *
t168 * t157 + 0.2e1 * t161 * t174 + 0.2e1 * t168 * t174 + t94 *
t93;
t409 = t102 * t129 + t102 * t149 + t108 * t129 + t108 * t149 +
t112 * t142 + t117 * t142 + t130 * t129 + t94 * t129 + 0.2e1 *
t161 * t179 + 0.2e1 * t161 * t185 + 0.2e1 * t168 * t179 + 0.2e1 *
t168 * t185;
t411 = 0.1e1 / (t392 + t409);
t425 = (-t112 * t80 - t117 * t80 - t35 * t207 - t360 * t209 -
t362 * t209 - t371 * t26 + t366 * t57 - t44 * t80 + t49 * t57 -
t65 * t88) * t11;
t435 = t28 * t5;
t444 = t21 * t28;
t451 = t102 * t21 + t102 * t5 + t108 * t21 + t108 * t5 + t112 *
t29 + t117 * t29 + t130 * t5 + 0.2e1 * t161 * t435 + 0.2e1 * t161
* t444 + 0.2e1 * t168 * t435 + 0.2e1 * t168 * t444 + t94 * t5;
t453 = (-t12 * t20 * t15 - t12 * t4 * t7 + t259 + t266 + t285 +
t290) / t451;
t = -t375 * t411 * r11 - t425 * t411 * r21 + t453 * r31;
u = -t375 * t411 * r12 - t425 * t411 * r22 + t453 * r32;
v = -t375 * t411 * r13 - t425 * t411 * r23 + t453 * r33;
```

A.5 Maple Optimization for Inverse Kinematics II

```
restart;
with(linalg);
```

```
[BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp, Wronskian, addcol,
    addrow, adj, adjoint, angle, augment, backsub, band, basis, bezout, blockmatrix, charmat,
    charpoly, cholesky, col, coldim, colspace, colspan, companion, concat, cond, copyinto,
    crossprod, curl, definite, delcols, delrows, det, diag, diverge, dotprod, eigenvals, eigenvalues,
    eigenvectors, eigenvects, entermatrix, equal, exponential, extend, ffgausselim, fibonacci,
    forwardsub, frobenius, gausselim, gaussjord, geneqns, genmatrix, grad, hadamard, hermite,
    hessian, hilbert, htranspose, ihermite, indexfunc, innerprod, intbasis, inverse, ismith,
    issimilar, iszero, jacobian, jordan, kernel, laplacian, leastsqrs, linsolve, matadd, matrix,
    minor, minpoly, mulcol, mulrow, multiply, norm, normalize, nullspace, orthog, permanent,
    pivot, potential, randmatrix, randvector, rank, ratform, row, rowdim, rowspace, rowspan,
    rref, scalarmul, singularvals, smith, stackmatrix, submatrix, subvector, sumbasis, swapcol,
    swaprow, sylvester, toeplitz, trace, transpose, vandermonde, vecpotent, vectdim, vector,
    wronskian]
```

(1)

```
with(CodeGeneration);
```

```
[C, CSharp, Fortran, IntermediateCode, Java, JavaScript, LanguageDefinition, Matlab, Names,
    Perl, Python, R, Save, Translate, VisualBasic]
```

(2)

```
RotZ(q) := matrix(4, 4, [[cos(q), -sin(q), 0, 0], [sin(q), cos(q), 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]) :
RotX(q) := matrix(4, 4, [[1, 0, 0, 0], [0, cos(q), -sin(q), 0], [0, sin(q), cos(q), 0], [0, 0, 0, 1]]) :
Trans(X, Y, Z) := matrix(4, 4, [[1, 0, 0, X], [0, 1, 0, Y], [0, 0, 1, Z], [0, 0, 0, 1]]) :
T1 := multiply(RotZ(q1), Trans(0, 0, L1), Trans(L2, 0, 0), RotX(-pi/2)) :
T2 := multiply(RotZ(q2 - pi/2), Trans(L3, 0, 0)) :
T3 := multiply(RotZ(q3), Trans(0, 0, L4), RotX(-pi/2)) :
T4 := Trans(L5, 0, 0) :
T04 := multiply(T1, T2, T3, T4) :
R04 := submatrix(T04, 1..3, 1..3) :
R04i := inverse(R04) :
R0 := matrix(3, 3, [[r11, r12, r13], [r21, r22, r23], [r31, r32, r33]])
```

$$\begin{bmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{bmatrix} \tag{3}$$

```
TA := multiply(R04i, R0) :
cs := [r = TA[1, 3], s = TA[2, 3], t = TA[3, 1], u = TA[3, 2], v = TA[3, 3]] :
Matlab(cs, optimize)
```

```
t1 = pi / 0.2e1;
t2 = -q2 + t1;
t3 = cos(t2);
t4 = cos(q1);
t5 = t3 * t4;
t6 = cos(t1);
t7 = t6 ^ 2;
t8 = cos(q3);
t9 = t7 * t8;
t10 = t5 * t9;
t11 = sin(t1);
t12 = t11 ^ 2;
```

```
t13 = t8 * t12;
t15 = t4 * t7;
t16 = sin(q3);
t17 = sin(t2);
t18 = t16 * t17;
t19 = t15 * t18;
t20 = t4 * t17;
t21 = t16 * t12;
t23 = sin(q1);
t24 = t23 * t3;
t25 = t6 * t16;
t27 = t23 * t6;
t28 = t8 * t17;
t31 = t23 ^ 2;
t32 = t3 ^ 2;
t33 = t31 * t32;
t34 = t16 ^ 2;
t35 = t7 * t34;
t37 = t8 ^ 2;
t38 = t7 * t37;
t40 = t34 * t12;
t42 = t37 * t12;
t44 = t31 * t7;
t45 = t17 ^ 2;
t46 = t34 * t45;
t48 = t37 * t45;
t50 = t31 * t45;
t53 = t4 ^ 2;
t54 = t32 * t53;
t59 = t53 * t7;
t62 = t53 * t45;
t65 = t33 * t35 + t33 * t38 + t33 * t40 + t33 * t42 + t54 * t35 +
t54 * t38 + t50 * t40 + t54 * t40 + t62 * t40 + t50 * t42 + t54 *
t42 + t62 * t42 + t44 * t46 + t44 * t48 + t59 * t46 + t59 * t48;
t66 = 0.1e1 / t65;
t69 = t24 * t9;
t71 = t23 * t7;
t72 = t71 * t18;
t73 = t23 * t17;
t76 = t4 * t6;
t84 = t32 * t7;
t85 = t84 * t34;
t86 = t84 * t37;
t87 = t32 * t34;
t89 = t32 * t37;
t91 = t7 * t45;
t92 = t91 * t34;
t93 = t91 * t37;
t94 = t46 * t12;
t95 = t48 * t12;
r = -(-t5 * t13 - t20 * t21 + t24 * t25 - t27 * t28 - t10 - t19)
* t66 * r13 + (t24 * t13 + t73 * t21 + t5 * t25 - t76 * t28 + t69
+ t72) * t66 * r23 - (t16 * t3 - t28) * t11 / (t87 * t12 + t89 *
t12 + t85 + t86 + t92 + t93 + t94 + t95) * r33;
t100 = t23 * t32;
t103 = t23 * t45;
t106 = t7 * t6;
```

```
t107 = t106 * t16;
t109 = t25 * t12;
t113 = t28 * t12;
t116 = t7 ^ 2;
t117 = t116 * t34;
t119 = t116 * t37;
t121 = t35 * t12;
t124 = t38 * t12;
t127 = t12 ^ 2;
t128 = t34 * t127;
t130 = t37 * t127;
t132 = t31 * t116;
t141 = t33 * t117 + t33 * t119 + 0.2e1 * t33 * t121 + 0.2e1 * t33
* t124 + t33 * t128 + t50 * t128 + t33 * t130 + t50 * t130 + t132
* t46 + t132 * t48 + 0.2e1 * t44 * t94 + 0.2e1 * t44 * t95;
t150 = t53 * t116;
t159 = t54 * t117 + t54 * t119 + 0.2e1 * t54 * t121 + 0.2e1 * t54
* t124 + t54 * t128 + t62 * t128 + t54 * t130 + t62 * t130 + t150
* t46 + t150 * t48 + 0.2e1 * t59 * t94 + 0.2e1 * t59 * t95;
t161 = 0.1e1 / (t141 + t159);
t164 = t32 * t4;
t167 = t4 * t45;
t197 = t46 * t116 + t48 * t116 + t117 * t32 + t119 * t32 + t128 *
t32 + t128 * t45 + t130 * t32 + t130 * t45 + 0.2e1 * t40 * t84 +
0.2e1 * t40 * t91 + 0.2e1 * t42 * t84 + 0.2e1 * t42 * t91;
t198 = 0.1e1 / t197;
s = (t4 * t106 * t28 + t100 * t40 + t100 * t42 + t103 * t40 +
t103 * t42 - t5 * t107 - t5 * t109 + t76 * t113 - t69 - t72) *
t161 * r13 - (-t23 * t106 * t28 + t24 * t107 + t24 * t109 - t27 *
t113 + t164 * t40 + t164 * t42 + t167 * t40 + t167 * t42 - t10 -
t19) * t161 * r23 - (t8 * t3 + t18 + t46 + t48 + t87 + t89) * t6
* t11 * t198 * r33;
t202 = t6 * t34;
t204 = t6 * t37;
t208 = t7 * t16;
t213 = t6 * t8;
t217 = t11 * (t100 * t202 + t100 * t204 - t20 * t13 - t15 * t28 +
t27 * t18 + t5 * t208 + t5 * t21 + t24 * t213 + t27 * t46 + t27 *
t48);
t231 = t11 * (-t73 * t13 - t164 * t202 - t164 * t204 - t76 * t18
+ t24 * t208 + t24 * t21 - t5 * t213 - t71 * t28 - t76 * t46 -
t76 * t48);
t239 = (-t12 * t17 * t16 - t12 * t3 * t8 + t85 + t86 + t92 + t93)
* t198;
t = -t217 * t161 * r11 - t231 * t161 * r21 + t239 * r31;
u = -t217 * t161 * r12 - t231 * t161 * r22 + t239 * r32;
v = -t217 * t161 * r13 - t231 * t161 * r23 + t239 * r33;
```

A.6 Calculation Jacobian Matrix Symbolic

```
%Initialize
clear;
clc;
close all;

% Symbolic
syms q1 q2 q3 q4 q5 q6 L1 L2 L3 L4 L5 L6 L7 L8 LtoolX LtoolZ;
%syms q1 q2 q3 q4 q5 q6;

% Values If not symbolic

% L1=0.78;
% L2=0.320;
% L3=1.075;
% L4=0.200;
% L5=1.142;
% L6=0.200;
% LtoolX=0;
% LtoolZ=0;

% Angles
% q1=0;
% q2=0;
% q3=0;
% q4=0;
% q5=pi/2;
% q6=0;

% Computing symbolic transformation matrixes from DH data

T01= RotZ(q1);
T01b = TrZ(L2)*TrX(L1)*RotX(-sym(pi/2));
T02= T01 * T01b * RotZ(q2);
T02b = RotZ(-sym(pi/2))*TrX(L3);
T03= T02 * T02b * RotZ(q3);
T03b = TrX(L4)*RotX(-sym(pi/2));
T04= T03 * T03b * RotZ(q4);
T04b = TrZ(L5)*RotX(sym(pi/2));
T05= T04 * T04b * RotZ(q5);
T05b = RotX(-sym(pi/2));
T06= T05 * T05b * RotZ(q6);
T06b = RotZ(sym(pi))*TrZ(L6+LtoolZ)*TrX(LtoolX);
T07 = T06 * T06b;

% Extracting rotation and position for each joint

R_00=T01(1:3,3);
o_00=T01(1:3,4);

R_01=T02(1:3,3);
o_01=T02(1:3,4);

R_02=T03(1:3,3);
o_02=T03(1:3,4);

R_03=T04(1:3,3);
o_03=T04(1:3,4);

R_04=T05(1:3,3);
```

```
o_04=T05(1:3,4);

R_05=T06(1:3,3);
o_05=T06(1:3,4);

o_06=T07(1:3,4);

% calculating the coloums in the jacobian matrix

J1=[cross(R_00,o_06-o_00);R_00];
J2=[cross(R_01,o_06-o_01);R_01];
J3=[cross(R_02,o_06-o_02);R_02];
J4=[cross(R_03,o_06-o_03);R_03];
J5=[cross(R_04,o_06-o_04);R_04];
J6=[cross(R_05,o_06-o_05);R_05];

% Jacobian matrix
J=[J1 J2 J3 J4 J5 J6];

%Traditional way to calculate Jacobian
%  J_t=jacobian([T07(1,4);T07(2,4);T07(3,4)],[q1;q2;q3]);
%  J_t(4:6,4:6)=zeros(3,3);
```

Published with MATLAB® R2017a

A.7 Jacobian Matrix Symbolic

Contents

Calculated symbolic Jacobian Matrix

```
  J = [J11 J12 J13 J14 J15 J16;...
       J21 J22 J23 J24 J25 J26;...
       J31 J32 J33 J34 J35 J36;...
       J41 J42 J43 J44 J45 J46;...
       J51 J52 J53 J54 J55 J56;...
       J61 J62 J63 J64 J65 J66];

% where:


J11 =(cos(q5)*(sin(q1)*sin(q2)*sin(q3) - cos(q2)*cos(q3)*sin(q1)))/...
     5 - (43*sin(q1)*sin(q2))/40 - (sin(q5)*(cos(q1)*sin(q4) - cos(q4)*...
     (cos(q2)*sin(q1)*sin(q3) + cos(q3)*sin(q1)*sin(q2))))/5 - ...
     (39*sin(q1))/50 + (571*sin(q1)*sin(q2)*sin(q3))/500 - (571*cos(q2)...
     *cos(q3)*sin(q1))/500 - (cos(q2)*sin(q1)*sin(q3))/5 - ...
     (cos(q3)*sin(q1)*sin(q2))/5;

J12 = -cos(q1)*((571*cos(q2)*sin(q3))/500 - (cos(q2)*cos(q3))/...
     5 - (43*cos(q2))/40 + (571*cos(q3)*sin(q2))/500 + (sin(q2)*...
     sin(q3))/5 + (cos(q5)*(cos(q2)*sin(q3) + cos(q3)*sin(q2)))/5 + ...
     (cos(q4)*sin(q5)*(cos(q2)*cos(q3) - sin(q2)*sin(q3)))/5);

J13 = -cos(q1)*((571*cos(q2)*sin(q3))/500 - (cos(q2)*cos(q3))/...
     5 + (571*cos(q3)*sin(q2))/500 + (sin(q2)*sin(q3))/5 + ...
     (cos(q5)*(cos(q2)*sin(q3) + cos(q3)*sin(q2)))/5 + ...
     (cos(q4)*sin(q5)*(cos(q2)*cos(q3) - sin(q2)*sin(q3)))/5);

J14 = (cos(q2)*sin(q3) + cos(q3)*sin(q2))*((sin(q5)*(cos(q1)*sin(q4)...
    - cos(q4)*(cos(q2)*sin(q1)*sin(q3) + cos(q3)*sin(q1)*sin(q2))))/...
    5 - (cos(q5)*(sin(q1)*sin(q2)*sin(q3) - cos(q2)*cos(q3)*sin(q1)))/...
    5 - (571*sin(q1)*sin(q2)*sin(q3))/500 + (571*cos(q2)*cos(q3)...
    *sin(q1))/500) + (sin(q1)*sin(q2)*sin(q3) - cos(q2)*cos(q3)...
    *sin(q1))*((571*cos(q2)*sin(q3))/500 + (571*cos(q3)*sin(q2))/...
    500 + (cos(q5)*(cos(q2)*sin(q3) + cos(q3)*sin(q2)))/5 + ...
    (cos(q4)*sin(q5)*(cos(q2)*cos(q3) - sin(q2)*sin(q3)))/5);

J15 = - (cos(q1)*cos(q4) + sin(q4)*(cos(q2)*sin(q1)*sin(q3) + ...
    cos(q3)*sin(q1)*sin(q2)))*((cos(q5)*(cos(q2)*sin(q3) + cos(q3)...
    *sin(q2)))/5 + (cos(q4)*sin(q5)*(cos(q2)*cos(q3) - sin(q2)*...
    sin(q3)))/5) - sin(q4)*((sin(q5)*(cos(q1)*sin(q4) - cos(q4)*...
    (cos(q2)*sin(q1)*sin(q3) + cos(q3)*sin(q1)*sin(q2))))/5 - ...
    (cos(q5)*(sin(q1)*sin(q2)*sin(q3) - cos(q2)*cos(q3)*sin(q1)))/5)*...
    (cos(q2)*cos(q3) - sin(q2)*sin(q3));

J16 = ((sin(q5)*(cos(q1)*sin(q4) - cos(q4)*(cos(q2)*sin(q1)*sin(q3)...
    + cos(q3)*sin(q1)*sin(q2))))/5 - (cos(q5)*(sin(q1)*sin(q2)*sin(q3)...
    - cos(q2)*cos(q3)*sin(q1)))/5)*(cos(q5)*(cos(q2)*sin(q3) + cos(q3)...
    *sin(q2)) + cos(q4)*sin(q5)*(cos(q2)*cos(q3) - sin(q2)*sin(q3)))...
    - (sin(q5)*(cos(q1)*sin(q4) - cos(q4)*(cos(q2)*sin(q1)*sin(q3)...
    + cos(q3)*sin(q1)*sin(q2))) - cos(q5)*(sin(q1)*sin(q2)*sin(q3)...
    - cos(q2)*cos(q3)*sin(q1)))*((cos(q5)*(cos(q2)*sin(q3) + cos(q3)...
   *sin(q2)))/5 + (cos(q4)*sin(q5)*(cos(q2)*cos(q3) - sin(q2)*sin(q3)))/5);
```

```
J21 = (39*cos(q1))/50 + (43*cos(q1)*sin(q2))/40 - (sin(q5)*(sin(q1)...
    *sin(q4) + cos(q4)*(cos(q1)*cos(q2)*sin(q3) + cos(q1)*cos(q3)...
    *sin(q2))))/5 + (cos(q5)*(cos(q1)*cos(q2)*cos(q3) - cos(q1)...
    *sin(q2)*sin(q3)))/5 + (571*cos(q1)*cos(q2)*cos(q3))/500 + ...
    (cos(q1)*cos(q2)*sin(q3))/5 + (cos(q1)*cos(q3)*sin(q2))/...
    5 - (571*cos(q1)*sin(q2)*sin(q3))/500;

J22 = -sin(q1)*((571*cos(q2)*sin(q3))/500 - (cos(q2)*cos(q3))/...
    5 - (43*cos(q2))/40 + (571*cos(q3)*sin(q2))/500 + (sin(q2)...
    *sin(q3))/5 + (cos(q5)*(cos(q2)*sin(q3) + cos(q3)*sin(q2)))/...
    5 + (cos(q4)*sin(q5)*(cos(q2)*cos(q3) - sin(q2)*sin(q3)))/5);

J23 = -sin(q1)*((571*cos(q2)*sin(q3))/500 - (cos(q2)*cos(q3))/...
    5 + (571*cos(q3)*sin(q2))/500 + (sin(q2)*sin(q3))/5 + (cos(q5)...
    *(cos(q2)*sin(q3) + cos(q3)*sin(q2)))/5 + (cos(q4)*sin(q5)...
    *(cos(q2)*cos(q3) - sin(q2)*sin(q3)))/5);

J24 = (cos(q1)*cos(q2)*cos(q3) - cos(q1)*sin(q2)*sin(q3))...
    *((571*cos(q2)*sin(q3))/500 + (571*cos(q3)*sin(q2))/...
    500 + (cos(q5)*(cos(q2)*sin(q3) + cos(q3)*sin(q2)))/...
    5 + (cos(q4)*sin(q5)*(cos(q2)*cos(q3) - sin(q2)*sin(q3)))/5)...
    + (cos(q2)*sin(q3) + cos(q3)*sin(q2))*((sin(q5)*(sin(q1)*sin(q4)...
    + cos(q4)*(cos(q1)*cos(q2)*sin(q3) + cos(q1)*cos(q3)*sin(q2))))/...
    5 - (cos(q5)*(cos(q1)*cos(q2)*cos(q3) - cos(q1)*sin(q2)...
    *sin(q3)))/5 - (571*cos(q1)*cos(q2)*cos(q3))/...
    500 + (571*cos(q1)*sin(q2)*sin(q3))/500);

J25 = - ((cos(q5)*(cos(q2)*sin(q3) + cos(q3)*sin(q2)))/5 + ...
    (cos(q4)*sin(q5)*(cos(q2)*cos(q3) - sin(q2)*sin(q3)))/5)...
    *(cos(q4)*sin(q1) - sin(q4)*(cos(q1)*cos(q2)*sin(q3) + ...
    cos(q1)*cos(q3)*sin(q2))) - sin(q4)*((sin(q5)*(sin(q1)*...
    sin(q4) + cos(q4)*(cos(q1)*cos(q2)*sin(q3) + cos(q1)*cos(q3)...
    *sin(q2))))/5 - (cos(q5)*(cos(q1)*cos(q2)*cos(q3) - cos(q1)...
    *sin(q2)*sin(q3)))/5)*(cos(q2)*cos(q3) - sin(q2)*sin(q3));

J26 = ((sin(q5)*(sin(q1)*sin(q4) + cos(q4)*(cos(q1)*cos(q2)*sin(q3)...
    + cos(q1)*cos(q3)*sin(q2))))/5 - (cos(q5)*(cos(q1)*cos(q2)...
    *cos(q3) - cos(q1)*sin(q2)*sin(q3)))/5)*(cos(q5)*(cos(q2)*sin(q3)...
    + cos(q3)*sin(q2)) + cos(q4)*sin(q5)*(cos(q2)*cos(q3) - sin(q2)...
    *sin(q3))) - (sin(q5)*(sin(q1)*sin(q4) + cos(q4)*(cos(q1)...
    *cos(q2)*sin(q3) + cos(q1)*cos(q3)*sin(q2))) - cos(q5)*(cos(q1)...
    *cos(q2)*cos(q3) - cos(q1)*sin(q2)*sin(q3)))*((cos(q5)...
    *(cos(q2)*sin(q3)+ cos(q3)*sin(q2)))/5 + (cos(q4)*sin(q5)...
    *(cos(q2)*cos(q3) - sin(q2)*sin(q3)))/5);

J31 = 0;

J32 = - cos(q1)*((43*cos(q1)*sin(q2))/40 - (sin(q5)*(sin(q1)*sin(q4)...
    + cos(q4)*(cos(q1)*cos(q2)*sin(q3) + cos(q1)*cos(q3)*sin(q2))))...
    /5+ (cos(q5)*(cos(q1)*cos(q2)*cos(q3) - cos(q1)*sin(q2)*sin(q3)))...
    /5 + (571*cos(q1)*cos(q2)*cos(q3))/500 + (cos(q1)*cos(q2)*sin(q3))...
    /5 + (cos(q1)*cos(q3)*sin(q2))/5 - (571*cos(q1)*sin(q2)*sin(q3))...
    /500) - sin(q1)*((43*sin(q1)*sin(q2))/40 + (sin(q5)*(cos(q1)...
    *sin(q4) - cos(q4)*(cos(q2)*sin(q1)*sin(q3) + cos(q3)*...
    sin(q1)*sin(q2))))/5 - (cos(q5)*(sin(q1)*sin(q2)*sin(q3) - cos(q2)...
    *cos(q3)*sin(q1)))/5 - (571*sin(q1)*sin(q2)*sin(q3))/500 + ...
    (571*cos(q2)*cos(q3)*sin(q1))/500 + (cos(q2)*sin(q1)*sin(q3))/...
    5 + (cos(q3)*sin(q1)*sin(q2))/5);

J33 = - sin(q1)*((sin(q5)*(cos(q1)*sin(q4) - cos(q4)*(cos(q2)*...
```

```
    sin(q1)*sin(q3) + cos(q3)*sin(q1)*sin(q2))))/5 - (cos(q5)*...
    (sin(q1)*sin(q2)*sin(q3) - cos(q2)*cos(q3)*sin(q1)))/5 - ...
    (571*sin(q1)*sin(q2)*sin(q3))/500 + (571*cos(q2)*cos(q3)*sin(q1))/...
    500 + (cos(q2)*sin(q1)*sin(q3))/5 + (cos(q3)*sin(q1)*sin(q2))/5)...
    - cos(q1)*((cos(q5)*(cos(q1)*cos(q2)*cos(q3) - cos(q1)*sin(q2)...
    *sin(q3)))/5 - (sin(q5)*(sin(q1)*sin(q4) + cos(q4)*(cos(q1)...
    *cos(q2)*sin(q3) + cos(q1)*cos(q3)*sin(q2))))/5 + (571*cos(q1)...
    *cos(q2)*cos(q3))/500 + (cos(q1)*cos(q2)*sin(q3))/5 + (cos(q1)...
    *cos(q3)*sin(q2))/5 - (571*cos(q1)*sin(q2)*sin(q3))/500);

J34 = (cos(q1)*cos(q2)*cos(q3) - cos(q1)*sin(q2)*sin(q3))*...
    ((sin(q5)*(cos(q1)*sin(q4) - cos(q4)*(cos(q2)*sin(q1)*...
    sin(q3) + cos(q3)*sin(q1)*sin(q2))))/5 - (cos(q5)*...
    (sin(q1)*sin(q2)*sin(q3) - cos(q2)*cos(q3)*sin(q1)))/...
    5 - (571*sin(q1)*sin(q2)*sin(q3))/500 + (571*cos(q2)*cos(q3)...
    *sin(q1))/500) - (sin(q1)*sin(q2)*sin(q3) - cos(q2)*cos(q3)...
    *sin(q1))*((sin(q5)*(sin(q1)*sin(q4) + cos(q4)*(cos(q1)*...
    cos(q2)*sin(q3) + cos(q1)*cos(q3)*sin(q2))))/5 - (cos(q5)*...
    (cos(q1)*cos(q2)*cos(q3) - cos(q1)*sin(q2)*sin(q3)))/5 - ...
    (571*cos(q1)*cos(q2)*cos(q3))/500 + (571*cos(q1)*sin(q2)*sin(q3))/500);

J35 = (cos(q1)*cos(q4) + sin(q4)*(cos(q2)*sin(q1)*sin(q3) + cos(q3)...
    *sin(q1)*sin(q2)))*((sin(q5)*(sin(q1)*sin(q4) + cos(q4)*(cos(q1)...
    *cos(q2)*sin(q3) + cos(q1)*cos(q3)*sin(q2))))/5 - (cos(q5)*(cos(q1)...
    *cos(q2)*cos(q3) - cos(q1)*sin(q2)*sin(q3)))/5) - ((sin(q5)*...
    (cos(q1)*sin(q4) - cos(q4)*(cos(q2)*sin(q1)*sin(q3) + cos(q3)*...
    sin(q1)*sin(q2))))/5 - (cos(q5)*(sin(q1)*sin(q2)*sin(q3) - cos(q2)...
    *cos(q3)*sin(q1)))/5)*(cos(q4)*sin(q1) - sin(q4)*(cos(q1)*cos(q2)...
    *sin(q3) + cos(q1)*cos(q3)*sin(q2)));

J36 = ((sin(q5)*(sin(q1)*sin(q4) + cos(q4)*(cos(q1)*cos(q2)*sin(q3)...
    + cos(q1)*cos(q3)*sin(q2))))/5 - (cos(q5)*(cos(q1)*cos(q2)*cos(q3)...
    - cos(q1)*sin(q2)*sin(q3)))/5)*(sin(q5)*(cos(q1)*sin(q4) - cos(q4)...
    *(cos(q2)*sin(q1)*sin(q3) + cos(q3)*sin(q1)*sin(q2))) - cos(q5)...
    *(sin(q1)*sin(q2)*sin(q3) - cos(q2)*cos(q3)*sin(q1))) - (sin(q5)...
    *(sin(q1)*sin(q4) + cos(q4)*(cos(q1)*cos(q2)*sin(q3) + cos(q1)...
    *cos(q3)*sin(q2))) - cos(q5)*(cos(q1)*cos(q2)*cos(q3) - cos(q1)...
    *sin(q2)*sin(q3)))*((sin(q5)*(cos(q1)*sin(q4) - cos(q4)*(cos(q2)...
    *sin(q1)*sin(q3) + cos(q3)*sin(q1)*sin(q2))))/5 - (cos(q5)*...
    (sin(q1)*sin(q2)*sin(q3) - cos(q2)*cos(q3)*sin(q1)))/5);

J41 = 0;

J42 = -sin(q1);

J43 = -sin(q1);

J44 = cos(q1)*cos(q2)*cos(q3) - cos(q1)*sin(q2)*sin(q3);

J45 = sin(q4)*(cos(q1)*cos(q2)*sin(q3) + cos(q1)*cos(q3)*sin(q2))...
    - cos(q4)*sin(q1);

J46 = cos(q5)*(cos(q1)*cos(q2)*cos(q3) - cos(q1)*sin(q2)*sin(q3))...
    - sin(q5)*(sin(q1)*sin(q4) + cos(q4)*(cos(q1)*cos(q2)*sin(q3)...
    + cos(q1)*cos(q3)*sin(q2)));

J51 = 0;

J52 = cos(q1);

J53 = cos(q1);
```

```
J54 = cos(q2)*cos(q3)*sin(q1) - sin(q1)*sin(q2)*sin(q3);

J55 = cos(q1)*cos(q4) + sin(q4)*(cos(q2)*sin(q1)*sin(q3)...
    + cos(q3)*sin(q1)*sin(q2));

J56 = sin(q5)*(cos(q1)*sin(q4) - cos(q4)*(cos(q2)*sin(q1)*sin(q3)...
    + cos(q3)*sin(q1)*sin(q2))) - cos(q5)*(sin(q1)*sin(q2)*sin(q3)...
    - cos(q2)*cos(q3)*sin(q1));

J61 = 1;

J62 = 0;

J63 = 0;

J64 = - cos(q2)*sin(q3) - cos(q3)*sin(q2);

J65 = sin(q4)*(cos(q2)*cos(q3) - sin(q2)*sin(q3));

J66 = - cos(q5)*(cos(q2)*sin(q3) + cos(q3)*sin(q2))...
      - cos(q4)*sin(q5)*(cos(q2)*cos(q3) - sin(q2)*sin(q3));
```

Published with MATLAB® R2017a

A.8 Stiffness Identification Joint 1

```
clear all;
close all;
clc;

%From global coordinate origin to where force is applied
L1 = (200 + 180 + 1142 + 320)*1e-3;

%Distance from global coordinate origin to reflector position (radius)
L2 = sqrt(120^2+480^2)*1e-3;

%The load force data. Mean values already calculated in Excel.
F = [-126.2130 -122.8430 -8.6697 13.2552 -14.1647 6.1561];
%The force sensor is mounted with an offset.
R=RotZ(atan2(219.25,226.66));
P=R*[F(:,1)' ; F(:,2)' ; F(:,3)';zeros(length(F(:,1)),1)'];
F(:,1)=P(1,:)';
F(:,2)=P(2,:)';
F(:,3)=P(3,:)';
P=R*[F(:,4)' ; F(:,5)' ; F(:,6)';zeros(length(F(:,1)),1)'];
F(:,4)=P(1,:)';
F(:,5)=P(2,:)';
F(:,6)=P(3,:)';

Fx=F(1);
Fy=F(2);
Fz=F(3);

%Reflector position no load
load FARO_0a.txt;
P=FARO_0a;
t=P(:,1)-P(1,1);

X0=P(:,2);
Y0=P(:,3);
Z0=P(:,4);

i=find(t<40);
x0 = mean(X0(i));
y0 = mean(Y0(i));
z0 = mean(Z0(i));

%Reflector position load applied
load FARO_1a.txt;
P=FARO_1a;
t=P(:,1)-P(1,1);

X=P(:,2);
Y=P(:,3);
Z=P(:,4);

i=find(t<40);
x = mean(X(i));
y = mean(Y(i));
z = mean(Z(i));

F_load  = Fy;

L  = sqrt((x-x0)^2+(y-y0)^2+(z-z0)^2);
% stiffness
```

```
K1 = abs(F_load*L1*L2)/L;


% Measured total displacement reflector
dispref = L;

% Total displacement reflector from SimX simulation
disprefSimX = sqrt((6.405e-005)^2+(1.671e-005)^2+(0)^2);

% Deviation between measured and simulated displacement
deviation = abs(dispref - disprefSimX) * 1e6; % [micrometer]
```

Published with MATLAB® R2017a

A.9 Stiffness Identification Joint 2

```
%init

clear all;
close all;
clc;

% Reflector Puck radius
r1 = 25;
% Robot "ring" radius at joint 3
r2 = 50;
% Distance from joint origin to where force is applied
L1 = (1075)*1e-3;
% Distance from joint origin to reflector position
L2 = (1075+r2-r1)*1e-3;

% The load was not applied through the force sensor. The load 20.120 [Kg]
F  = 197; % [N]

%Reflector position no load applied
load FARO_0.txt;
P=FARO_0;
t=P(:,1)-P(1,1);

X0=P(:,2);
Y0=P(:,3);
Z0=P(:,4);

i=find(t<40);
x0 = mean(X0(i));
y0 = mean(Y0(i));
z0 = mean(Z0(i));

%Reflector position load applied
load FARO_1.txt;
P=FARO_1;
t=P(:,1)-P(1,1);

X=P(:,2);
Y=P(:,3);
Z=P(:,4);

i=find(t<40);
x = mean(X(i));
y = mean(Y(i));
z = mean(Z(i));

L  = sqrt((x-x0)^2+(y-y0)^2+(z-z0)^2);
% stiffness
K2 = (F*L1*L2)/L;

% Measured total displacement reflector
dispref = L;

% Total displacement reflector from SimX simulation
disprefSimX = sqrt((0.000108)^2+(5.163e-6)^2+(-2.82e-010)^2);

% Deviation between measured and simulated displacement
deviation = abs(dispref - disprefSimX) * 1e6; % [micrometer]
```

Published with MATLAB® R2017a

A.10 Stiffness Identification Joint 3

```
% init

clear all;
close all;
clc;

%Distance from joint origin 3 to reflector position
L2 = sqrt((1142-88)^2+(207)^2)*1e-3;


%Tool properties. In this case, where the load is applied
LToolX = 0;
LToolZ = 0.180;

%The load force data. Mean values already calculated in Excel.
F  = [-1.6108   -0.4682 197.9959        1.7874  -0.4346 -0.0256];
%The force sensor is mounted with an offset.
R=RotZ(atan2(219.25,226.66));
P=R*[F(:,1)' ; F(:,2)' ; F(:,3)';zeros(length(F(:,1)),1)'];
F(:,1)=P(1,:)';
F(:,2)=P(2,:)';
F(:,3)=P(3,:)';
P=R*[F(:,4)' ; F(:,5)' ; F(:,6)';zeros(length(F(:,1)),1)'];
F(:,4)=P(1,:)';
F(:,5)=P(2,:)';
F(:,6)=P(3,:)';

Fx=F(1);
Fy=F(2);
Fz=F(3);

%Reflector position no load
load FARO_0.txt;
P=FARO_0;
t=P(:,1)-P(1,1);

X0=P(:,2);
Y0=P(:,3);
Z0=P(:,4);

i=find(t<40);
x0 = mean(X0(i));
y0 = mean(Y0(i));
z0 = mean(Z0(i));

%Reflector position load applied
load FARO_1.txt;
P=FARO_1;
t=P(:,1)-P(1,1);

X=P(:,2);
Y=P(:,3);
Z=P(:,4);

i=find(t<40);
x = mean(X(i));
y = mean(Y(i));
z = mean(Z(i));
```

```
% Finding the Jacobian. Link lengths and angles are set in the getJa
% function, as shown in "Calculation Jacobian Matrix Symbolic"
bSymbolic=false;
J=getJa(bSymbolic,LToolX,LToolZ);
JT=J';

K2=xxx; % confidential

Torque=JT*[0;0;-Fz;0;0;0];

dq2 = Torque(2)/K2;
DQ2 = J*[0 dq2 0 0 0 0]';

L = sqrt((x-x0-DQ2(1))^2+(y-y0-DQ2(2))^2+(z-z0-DQ2(3))^2);

% STiffness
K3 = (Torque(3)*L2) / L;

% Measured total displacement reflector at end effector
dispEE = sqrt((x-x0)^2+(y-y0)^2+(z-z0)^2);

% Total displacement reflector at end effector from SimX simulation
dispEESimX = sqrt((2.825e-5)^2+(2.906e-7)^2+(0.0001431)^2);

% Deviation between measured and simulated displacement
deviation = (dispEE - dispEESimX) * 1e6; % [micrometer]
```

```
Undefined function or variable 'xxx'.

Error in Stiff3a (line 69)
K2=xxx; % confidential
```

Published with MATLAB® R2017a

A.11 Stiffness Identification Joint 4

```
% Init
clear all;
close all;
clc;

% Lengths described in the report
L1 = (200+180)*1e-3;
R4  = (190/2)*1e-3;
L2 = (270)*1e-3;

%The load force data. Mean values already calculated in Excel.
load Force;
F = [Force{1,1} Force{1,2} Force{1,3} Force{1,4} Force{1,5} Force{1,6}];
%The force sensor is mounted with an offset.
R=RotZ(atan2(219.25,226.66));
P=R*[F(:,1)' ; F(:,2)' ; F(:,3)';zeros(length(F(:,1)),1)'];
F(:,1)=P(1,:)';
F(:,2)=P(2,:)';
F(:,3)=P(3,:)';
P=R*[F(:,4)' ; F(:,5)' ; F(:,6)';zeros(length(F(:,1)),1)'];
F(:,4)=P(1,:)';
F(:,5)=P(2,:)';
F(:,6)=P(3,:)';

Fx=F(1);
Fy=F(2);
Fz=F(3);

% Joint 4 inner reflector position no load
load FARO_inner_0.txt;
P=FARO_inner_0;

t_inner_0=P(:,1)-P(1,1);

X_inner_0=P(:,2);
Y_inner_0=P(:,3);
Z_inner_0=P(:,4);

i=find(t_inner_0<40);
x_inner_0 = mean(X_inner_0(i));
y_inner_0 = mean(Y_inner_0(i));
z_inner_0 = mean(Z_inner_0(i));

% Joint 4 outer reflector position no load
load FARO_outer_0.txt;
P=FARO_outer_0;
t_outer_0=P(:,1)-P(1,1);

X_outer_0=P(:,2);
Y_outer_0=P(:,3);
Z_outer_0=P(:,4);

i=find(t_outer_0<40);
x_outer_0 = mean(X_outer_0(i));
y_outer_0 = mean(Y_outer_0(i));
z_outer_0 = mean(Z_outer_0(i));

% Joint 4 inner reflector position load applied
load FARO_inner_1.txt;
```

```
P=FARO_inner_1;
t_inner_1=P(:,1)-P(1,1);

X_inner_1=P(:,2);
Y_inner_1=P(:,3);
Z_inner_1=P(:,4);

i=find(t_inner_1<40);
x_inner_1 = mean(X_inner_1(i));
y_inner_1 = mean(Y_inner_1(i));
z_inner_1 = mean(Z_inner_1(i));

% Joint 4 outer reflector position load applied
load FARO_outer_1.txt;
P=FARO_outer_1;
t_outer_1=P(:,1)-P(1,1);

X_outer_1=P(:,2);
Y_outer_1=P(:,3);
Z_outer_1=P(:,4);

i=find(t_outer_1<40);
x_outer_1 = mean(X_outer_1(i));
y_outer_1 = mean(Y_outer_1(i));
z_outer_1 = mean(Z_outer_1(i));

% Creating vectors between inner and outer
% reflector no load, and load applied.

vec_unload = [x_outer_0-x_inner_0 y_outer_0-y_inner_0 z_outer_0-z_inner_0];
vec_load = [x_outer_1-x_inner_1 y_outer_1-y_inner_1 z_outer_1-z_inner_1];

abs_vec_unload  = sqrt(vec_unload(1)^2+vec_unload(2)^2+vec_unload(3)^2);
abs_vec_load    = sqrt(vec_load(1)^2+vec_load(2)^2+vec_load(3)^2);

% Calculating angle between vectors
dot_prod    = (vec_unload(1)*vec_load(1)+vec_unload(2)*vec_load(2)...
               +vec_unload(3)*vec_load(3));
phi         = acos(dot_prod/(abs_vec_unload*abs_vec_load));
phi_deg     = phi*180/pi;

z_diff      = z_inner_1-z_inner_0;
x_diff      = x_inner_1-x_inner_0;
y_diff      = y_inner_1-y_inner_0;
alpha       = abs(atan((sqrt(z_diff^2+x_diff^2+y_diff^2))/L2));

% The curvature c of joint 4
c = abs_vec_unload * tan(phi-alpha);

% The angular displacement in joint 4
theta = atan(c/R4);

% The stiffness
K4 = Fx * L1 / theta;

% from SimX simulation
thetaSimX = 0.0002677; %[rad]

angdev = thetaSimX - theta;
% radius from joint 4 to reflector position
Rref = 0.407; %[m]
```

```
%Deviation between measured and simulated displacement
Deviation = angdev * Rref * 1e6; %[micrometer]
```

Published with MATLAB® R2017a

A.12 Stiffness Identification Joint 5

```
% Init
clear all;
close all;
clc;

%The load force data. Mean values already calculated in Excel.
F = [131.3839 -143.2498 3.7887 11.3128 10.8667 0.1326];
%The force sensor is mounted with an offset.
R=RotZ(atan2(219.25,226.66));
P=R*[F(:,1)' ; F(:,2)' ; F(:,3)';zeros(length(F(:,1)),1)'];
F(:,1)=P(1,:)';
F(:,2)=P(2,:)';
F(:,3)=P(3,:)';
P=R*[F(:,4)' ; F(:,5)' ; F(:,6)';zeros(length(F(:,1)),1)'];
F(:,4)=P(1,:)';
F(:,5)=P(2,:)';
F(:,6)=P(3,:)';

Fx=F(1);
Fy=F(2);
Fz=F(3);

% Joint 2 position no load
load FARO_J2_0.txt;
P=FARO_J2_0;
t_J20=P(:,1)-P(1,1);

X_J20=P(:,2);
Y_J20=P(:,3);
Z_J20=P(:,4);

i=find(t_J20<40);
x_J20 = mean(X_J20(i));
y_J20 = mean(Y_J20(i));
z_J20 = mean(Z_J20(i));

% Joint 3 position no load
load FARO_J3_0.txt;
P=FARO_J3_0;
t_J30=P(:,1)-P(1,1);

X_J30=P(:,2);
Y_J30=P(:,3);
Z_J30=P(:,4);

i=find(t_J30<40);
x_J30 = mean(X_J30(i));
y_J30 = mean(Y_J30(i));
z_J30 = mean(Z_J30(i));

% Joint 5 position no load
load FARO_J5_0.txt;
P=FARO_J5_0;
t_J50=P(:,1)-P(1,1);

X_J50=P(:,2);
Y_J50=P(:,3);
Z_J50=P(:,4);
```

```
i=find(t_J50<40);
x_J50 = mean(X_J50(i));
y_J50 = mean(Y_J50(i));
z_J50 = mean(Z_J50(i));

% End Effector position no load
load FARO_EE_0.txt;
P=FARO_EE_0;
t_EE0=P(:,1)-P(1,1);

X_EE0=P(:,2);
Y_EE0=P(:,3);
Z_EE0=P(:,4);

i=find(t_EE0<40);
x_EE0 = mean(X_EE0(i));
y_EE0 = mean(Y_EE0(i));
z_EE0 = mean(Z_EE0(i));

% Joint 3 position load applied
load FARO_J3_1.txt;
P=FARO_J3_1;
t_J31=P(:,1)-P(1,1);

X_J31=P(:,2);
Y_J31=P(:,3);
Z_J31=P(:,4);

i=find(t_J31<40);
x_J31 = mean(X_J31(i));
y_J31 = mean(Y_J31(i));
z_J31 = mean(Z_J31(i));

% Joint 5 position load applied
load FARO_J5_1.txt;
P=FARO_J5_1;
t_J51=P(:,1)-P(1,1);

X_J51=P(:,2);
Y_J51=P(:,3);
Z_J51=P(:,4);

i=find(t_J51<40);
x_J51 = mean(X_J51(i));
y_J51 = mean(Y_J51(i));
z_J51 = mean(Z_J51(i));

% End Effector loaded position load applied
load FARO_EE_1.txt;
P=FARO_EE_1;
t_EE1=P(:,1)-P(1,1);

X_EE1=P(:,2);
Y_EE1=P(:,3);
Z_EE1=P(:,4);

i=find(t_EE1<40);
x_EE1 = mean(X_EE1(i));
y_EE1 = mean(Y_EE1(i));
z_EE1 = mean(Z_EE1(i));
```

```
% Crating vectors unloaded robot

vec_02unload = [x_J20 y_J20 z_J20];
vec_03unload = [x_J30 y_J30 z_J30];
vec_05unload = [x_J50 y_J50 z_J50];
vec_0Eunload = [x_EE0 y_EE0 z_EE0];

vec_23unload = vec_03unload - vec_02unload;
vec_35unload = vec_05unload - vec_03unload;
vec_5Eunload = vec_0Eunload - vec_05unload;

vec_25unload = vec_05unload - vec_02unload;

% Crating vectors loaded robot

vec_02load = [x_J20 y_J20 z_J20];
vec_03load = [x_J31 y_J31 z_J31];
vec_05load = [x_J51 y_J51 z_J51];
vec_0Eload = [x_EE1 y_EE1 z_EE1];

vec_23load = vec_03load - vec_02load;
vec_35load = vec_05load - vec_03load;
vec_5Eload = vec_0Eload - vec_05load;

vec_25load = vec_05load - vec_02load;

% calculating the angle between the vector 2-5 and 5-EE unloaded

% First vector
m = vec_25unload;
% Second vector
n = vec_5Eunload;
% Angle between vectors
dot_prod    = (m(1)*n(1)+m(2)*n(2)+m(3)*n(3));
abs_m       = sqrt(m(1)^2+m(2)^2+m(3)^2);
abs_n       = sqrt(n(1)^2+n(2)^2+n(3)^2);
ang_25_5E_u         = acos(dot_prod/(abs_m*abs_n));
ang_25_5E_u_deg     = ang_25_5E_u*180/pi;

% calculating the angle between the vector 2-5 and 5-EE loaded

% First vector
o = vec_25load;
% Second vector
p = vec_5Eload;
% Angle between vectors
dot_prod    = (o(1)*p(1)+o(2)*p(2)+o(3)*p(3));
abs_o       = sqrt(o(1)^2+o(2)^2+o(3)^2);
abs_p       = sqrt(p(1)^2+p(2)^2+p(3)^2);
ang_25_5E         = acos(dot_prod/(abs_o*abs_p));
ang_25_5E_deg     = ang_25_5E*180/pi;

% comparing vector 2-5 and 5-EE loaded/unloaded

theta = ang_25_5E - ang_25_5E_u;

% Moment arms
L1 = (200 + 180)*1e-3;

% Stiffness
```

```
K5 = Fx*L1/theta;

% Displacements

%Displacement reflector3
dr3 = sqrt((x_J31-x_J30)^2+(y_J31-y_J30)^2+(z_J31-z_J30)^2);

%Displacement reflector5
dr5 = sqrt((x_J51-x_J50)^2+(y_J51-y_J50)^2+(z_J51-z_J50)^2);

%Displacement reflectorEE
drEE = sqrt((x_EE1-x_EE0)^2+(y_EE1-y_EE0)^2+(z_EE1-z_EE0)^2);

% Reflector Puck radius
r1 = 25;

% Robot "ring" radius at joint 3
r2 = 50;

%Radius from joint 2 to reflector 3
RadRef2 = (1075+r2-r1)*1e-3;

% from SimX simulation
dispEESimX = sqrt((5.281e-5)^2+(2.06e-8)^2+(0.0005593)^2);
% Deviation between measured and simulated displacement [micrometer]
dev = (drEE - dispEESimX) * 1e6;
```

Published with MATLAB® R2017a

A.13 Stiffness Identification Joint 6

```
%init
clear all;
close all;
clc;

%Tool properties. In this case, where the load is applied
LToolX = 0;
LToolZ = 0.180;

%The load force data. Mean values already calculated in Excel.
F  = [123.7268 -128.4797 3.4454 6.7128 4.6608 -89.4632];
%The force sensor is mounted with an offset.
R=RotZ(atan2(219.25,226.66));
P=R*[F(:,1)' ; F(:,2)' ; F(:,3)';zeros(length(F(:,1)),1)'];
F(:,1)=P(1,:)';
F(:,2)=P(2,:)';
F(:,3)=P(3,:)';
P=R*[F(:,4)' ; F(:,5)' ; F(:,6)';zeros(length(F(:,1)),1)'];
F(:,4)=P(1,:)';
F(:,5)=P(2,:)';
F(:,6)=P(3,:)';

Fx=F(1);
Fy=F(2);
Fz=F(3);

%Reflector position no load
load FARO_0re.txt;
P=FARO_0re;
t=P(:,1)-P(1,1);
%plot(t);

X0=P(:,2);
Y0=P(:,3);
Z0=P(:,4);

i=find(t<40);
x0 = mean(X0(i));
y0 = mean(Y0(i));
z0 = mean(Z0(i));

%Reflector position load applied
load FARO_1re.txt;
P=FARO_1re;
t=P(:,1)-P(1,1);
%plot(t);

X=P(:,2);
Y=P(:,3);
Z=P(:,4);

i=find(t<40);
x = mean(X(i));
y = mean(Y(i));
z = mean(Z(i));

% Finding the Jacobian. Link lengths and angles are set in the getJa
% function, as shown in "Calculation Jacobian Matrix Symbolic"
bSymbolic=false;
```

```
J=getJa(bSymbolic,LToolX,LToolZ);
JT=J';

K1=xxx; % confidential

Torque=JT*F';

dq1 = Torque(1)/K1;
DQ1 = J*[dq1 0 0 0 0 0]';

Lwq1 = sqrt((x-x0-DQ1(1))^2+(y-y0-DQ1(2))^2+(z-z0-DQ1(3))^2);
drEE = sqrt((x-x0)^2+(y-y0)^2+(z-z0)^2);

%Offset from spindle centre to reflector location L2
R = 265e-3;
Tx6 =F(6);

K6 = abs(Tx6*R)/Lwq1;

%From SimX Simulation
DispSimX = sqrt((0.0004344)^2+(5.96e-5)^2+(0.0001779)^2);

Deviation = (Lwq1 - DispSimX) * 1e6;
Deviation1 = (drEE - DispSimX) * 1e6;
```

```
Undefined function or variable 'xxx'.

Error in Stiff6 (line 64)
K1=xxx; % confidential
```

Published with MATLAB® R2017a

A.14 Influence of Kc on Kx

```
% Symbolic for the anglels
syms q1 q2 q3 q4 q5 q6;

% Link Lengths
L1=0.78;
L2=0.320;
L3=1.075;
L4=0.200;
L5=1.142;
L6=0.200;
LToolX=0;
LToolZ=0;

%Wrench
w = [0 0 -2000 200 200 0]; %[N N N Nm Nm Nm]

% Finding the Jacobian
Jsym = getJacobi(L1,L2,L3,L4,L5,L6,LToolX,LToolZ,q1,q2,q3,q4,q5,q6);

%Finding Kc
J1 = diff(Jsym',q1)*w';
J2 = diff(Jsym',q2)*w';
J3 = diff(Jsym',q3)*w';
J4 = diff(Jsym',q4)*w';
J5 = diff(Jsym',q5)*w';
J6 = diff(Jsym',q6)*w';

Kc = [J1 J2 J3 J4 J5 J6];

% Influence of Kc on Kx is then evaluated

% Stiffness values
K_th        = eye(6);
K_vec       = [x x x x x x]; %Confidential
K_th(1,1)   = K_vec(1,1);
K_th(2,2)   = K_vec(1,2);
K_th(3,3)   = K_vec(1,3);
K_th(4,4)   = K_vec(1,4);
K_th(5,5)   = K_vec(1,5);
K_th(6,6)   = K_vec(1,6);

% Now we want to lock these angles.
q1  = 0;
q4  = pi/4;
q5  = pi/4;
q6  = pi/4;

% But still symbolic for q2 and q3
syms q2 q3;

% range q3 radians
A = [-pi:0.007:1.2217];
B = zeros(1,length(A));
% range q2 radians
D = [-1.1345:0.007:1.4835];
E = zeros(length(A),length(D));
F = zeros(length(A),length(D));
H = zeros(length(A),length(D));
for i = 1:length(A) % Joint angle 3
```

```
    q3 = A(i);

    for j = 1:length(D) % Joint angle 2

        q2 = D(j);
        J = getJacobi(L1,L2,L3,L4,L5,L6,LToolX,LToolZ,q1,q2,q3,q4,q5,q6);

        %Kc = 0
        ddnull = J*((K_th)^(-1))*J'*w';
        dpnull = sqrt((ddnull(1)^2)+(ddnull(2)^2)+(ddnull(3)^2));

        %Kc is not 0
        dd = J*((K_th-Kc)^(-1))*J'*w';
        dp = sqrt((dd(1)^2)+(dd(2)^2)+(dd(3)^2));

        vpn = abs(dp-dpnull)/max(dp,dpnull);
        vp  = vpn;
        E(i,j) = vp;

        ddrx = abs(dd(4) - ddnull(4));
        ddry = abs(dd(5) - ddnull(5));
        ddrz = abs(dd(6) - ddnull(6));

        vrn = max([ddrx ddry ddrz]);
        vr  = vrn*180/pi;
        F(i,j) = vr;
        H(i,j) = dd(4);

    end
end

figure;
subplot(1,2,1);
contour(E,A,'ShowText','on');


x_label_beg = D(1)*180/pi;
x_label_end = D(end)*180/pi;

xtick_label_cellarr = [-65:20:85];
xtick_cellarr = linspace(0,length(D),numel(xtick_label_cellarr));
set(gca, 'XTick',xtick_cellarr);
set(gca, 'XTickLabel',xtick_label_cellarr);

y_label_beg = A(1)*180/pi;
y_label_end = A(end)*180/pi;

ytick_label_cellarr = [-180:20:70];
ytick_cellarr = linspace(0,length(A),numel(ytick_label_cellarr));
set(gca, 'YTick',ytick_cellarr);
set(gca, 'YTickLabel',ytick_label_cellarr);

title('Vp Influence of Kc on Kx [m]');
xlabel('Joint 2 [deg]');
ylabel('Joint 3 [deg]');

subplot(1,2,2);
contour(F,A,'ShowText','on');
colormap(winter);
```

```
x_label_beg = D(1)*180/pi;
x_label_end = D(end)*180/pi;

xtick_label_cellarr = [-65:20:85];
xtick_cellarr = linspace(0,length(D),numel(xtick_label_cellarr));
set(gca, 'XTick',xtick_cellarr);
set(gca, 'XTickLabel',xtick_label_cellarr);

y_label_beg = A(1)*180/pi;
y_label_end = A(end)*180/pi;

ytick_label_cellarr = [-180:20:70];
ytick_cellarr = linspace(0,length(A),numel(ytick_label_cellarr));
set(gca, 'YTick',ytick_cellarr);
set(gca, 'YTickLabel',ytick_label_cellarr);

title('Vr Influence of Kc on Kx [deg]');
xlabel('Joint 2 [deg]');
ylabel('Joint 3 [deg]');
```

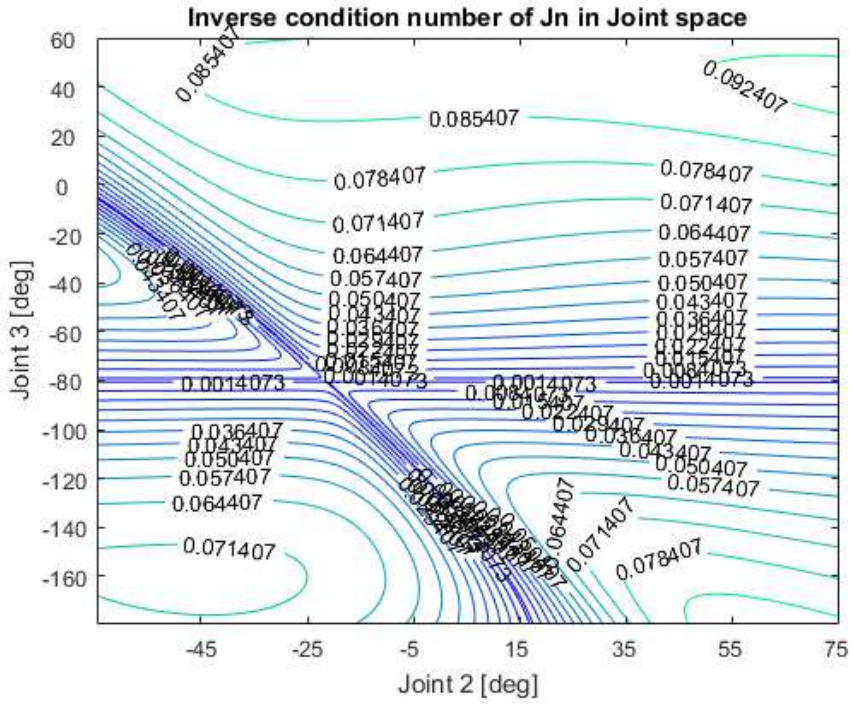
```
Undefined function or variable 'x'.

Error in KcClean (line 34)
K_vec       = [x x x x x]; %Confidential
```

Published with MATLAB® R2017a

A.15 Inverse Condition Number

```
%init
clear all;
close all;
clc;

%Link lengths
L1=0.78;
L2=0.320;
L3=1.075;
L4=0.200;
L5=1.142;
L6=0.200;
LToolX=0;
LToolZ=0;

%Angles
q1=0;
q4=pi/4;
q5=pi/4;
q6=pi/4;

set(0,'defaultfigurecolor',[1 1 1]);
% Range for Joint 2
A = [-pi:0.007:1.0472];
B = zeros(1,length(A));
% Range for Joint 3
D = [-1.1345:0.007:1.3963];
E = zeros(length(A),length(D));

for i = 1:length(A) % Joint angle 2

    q3 = A(i);

    for j = 1:length(D) % Joint angle 3

        q2 = D(j);

        J=getJacobi(L1,L2,L3,L4,L5,L6,LToolX,LToolZ,q1,q2,q3,q4,q5,q6);

        FrobNorm = cond(J,'fro');
        TwoNorm  = cond(J,2);
        FrobNormInv = inv(FrobNorm);
        TwoNormInv = inv(TwoNorm);
        E(i,j) = FrobNormInv;
    end
end



figure
contour(E,A,'ShowText','on');
colormap(winter)
x_label_beg = D(1)*180/pi;
x_label_end = D(end)*180/pi;

xtick_label_cellarr = [-65:20:80];
xtick_cellarr = linspace(0,length(D),numel(xtick_label_cellarr));
set(gca, 'XTick',xtick_cellarr);
set(gca, 'XTickLabel',xtick_label_cellarr);
```

```
y_label_beg = A(1)*180/pi;
y_label_end = A(end)*180/pi;

ytick_label_cellarr = [-180:20:60];
ytick_cellarr = linspace(0,length(A),numel(ytick_label_cellarr));
set(gca, 'YTick',ytick_cellarr);
set(gca, 'YTickLabel',ytick_label_cellarr);

title('Inverse condition number of Jn in Joint space')
xlabel('Joint 2 [deg]')
ylabel('Joint 3 [deg]')
```



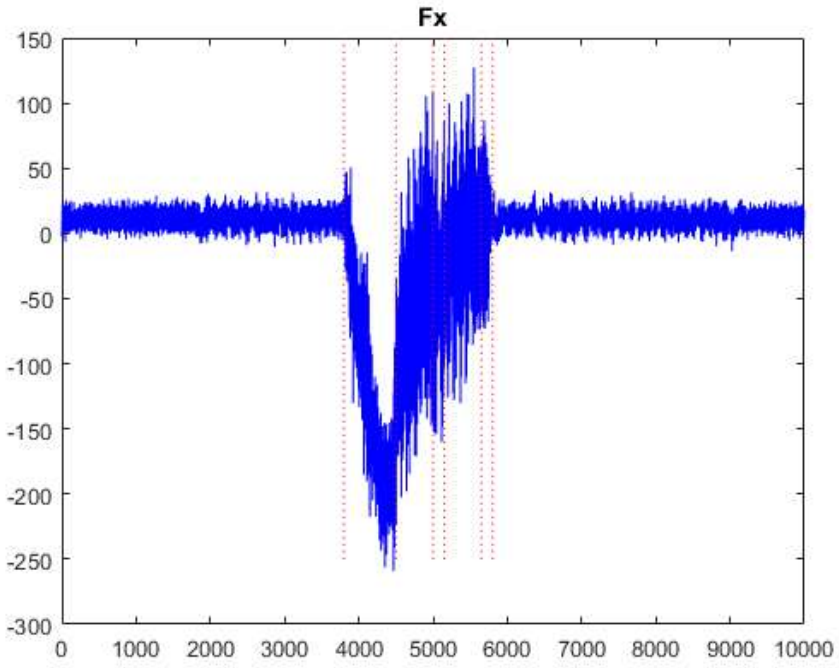
Published with MATLAB® R2017a

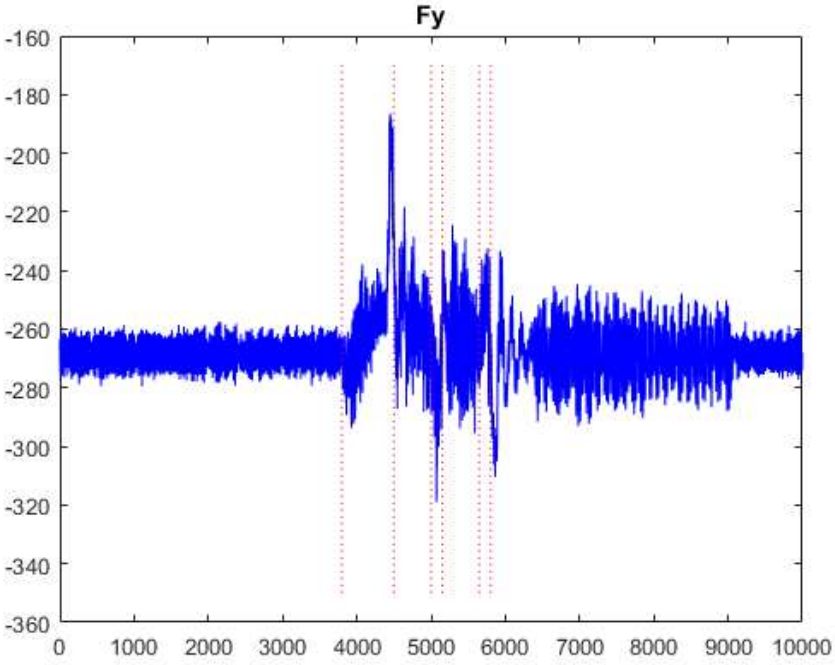
A.16 Process Force

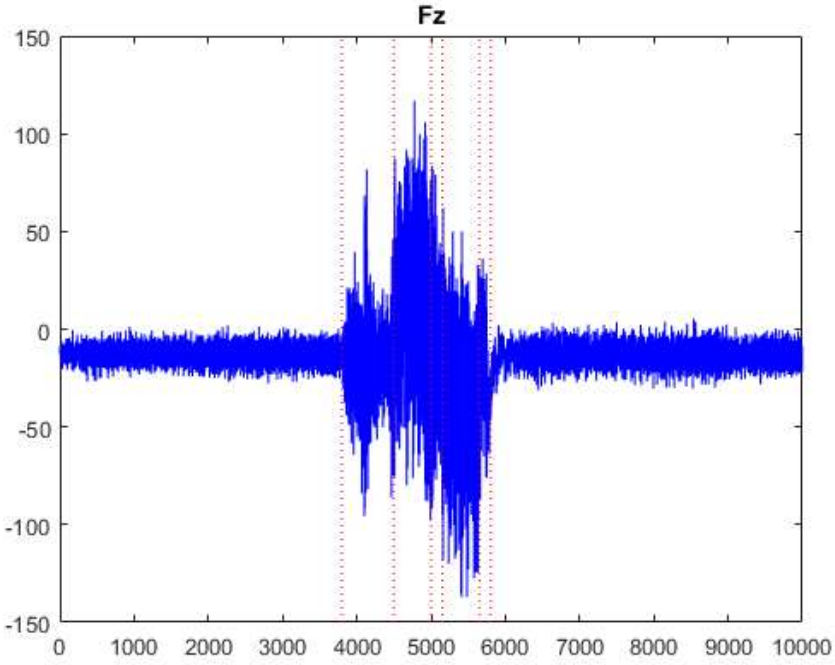
```
%init
clear all;
close all;
clc;

% Loading the process forces
load MillingForces1;
F1=MillingForces;
t=F1{:,11}-F1{1,11};
%The force sensor is mounted with an offset
R=RotZ(atan2(219.25,226.66));
P=R*[F1{:,1}' ; F1{:,2}' ; F1{:,3}';zeros(length(F1{:,1}),1)'];
F(:,1)=P(1,:)';
F(:,2)=P(2,:)';
F(:,3)=P(3,:)';
P=R*[F1{:,4}' ; F1{:,5}' ; F1{:,6}';zeros(length(F1{:,1}),1)'];
F(:,4)=P(1,:)';
F(:,5)=P(2,:)';
F(:,6)=P(3,:)';

Fx=F(:,1);
Fy=F(:,2);
Fz=F(:,3);
Mx=F(:,4);
My=F(:,5);
Mz=F(:,6);

% Setting the intervals
a = 4500;
b = 5000;
c = 5150;
d = 5650;

% Plotting the forces and moments
figure(1);
plot(t,Fx,'b')
hold on;
line([3800 3800], [-250 150],'LineStyle',':','Color','r');
line([5800 5800], [-250 150],'LineStyle',':','Color','r');
line([a a], [-250 150],'LineStyle',':','Color','r');
line([b b], [-250 150],'LineStyle',':','Color','r');
line([c c], [-250 150],'LineStyle',':','Color','r');
line([d d], [-250 150],'LineStyle',':','Color','r');
title('Fx')

figure(2);
plot(t,Fy,'b')
hold on;
line([3800 3800], [-350 -170],'LineStyle',':','Color','r');
line([5800 5800], [-350 -170],'LineStyle',':','Color','r');
line([a a], [-350 -170],'LineStyle',':','Color','r');
line([b b], [-350 -170],'LineStyle',':','Color','r');
line([c c], [-350 -170],'LineStyle',':','Color','r');
line([d d], [-350 -170],'LineStyle',':','Color','r');
title('Fy')

figure(3);
plot(t,Fz,'b')
hold on;
```

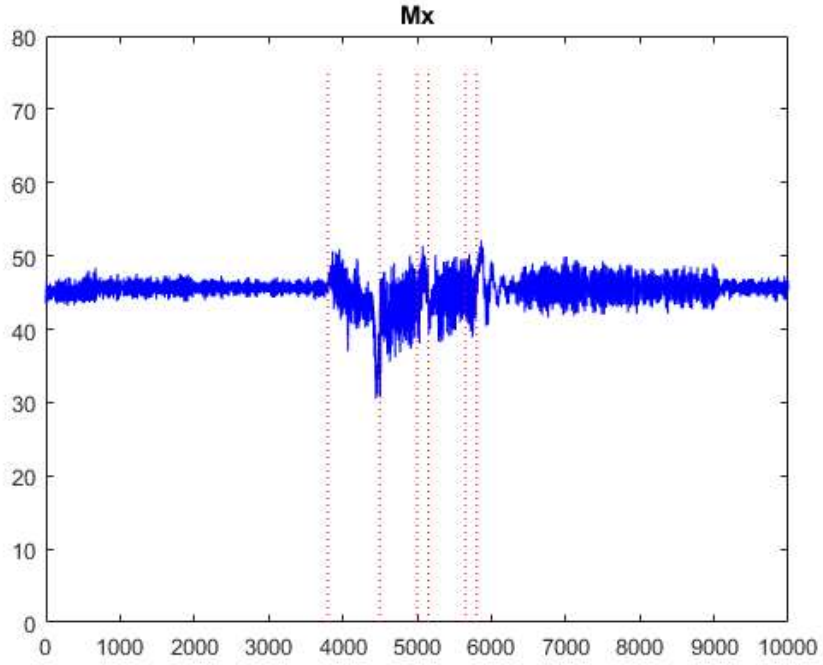
```
line([3800 3800], [-150 150],'LineStyle',':','Color','r');
line([5800 5800], [-150 150],'LineStyle',':','Color','r');
line([a a], [-150 150],'LineStyle',':','Color','r');
line([b b], [-150 150],'LineStyle',':','Color','r');
line([c c], [-150 150],'LineStyle',':','Color','r');
line([d d], [-150 150],'LineStyle',':','Color','r');
title('Fz')

figure(4);
plot(t,Mx,'b')
hold on;
line([3800 3800], [0 75],'LineStyle',':','Color','r');
line([5800 5800], [0 75],'LineStyle',':','Color','r');
line([a a], [0 75],'LineStyle',':','Color','r');
line([b b], [0 75],'LineStyle',':','Color','r');
line([c c], [0 75],'LineStyle',':','Color','r');
line([d d], [0 75],'LineStyle',':','Color','r');
title('Mx')

figure(5);
plot(t,My,'b')
hold on;
line([3800 3800], [-50 50],'LineStyle',':','Color','r');
line([5800 5800], [-50 50],'LineStyle',':','Color','r');
line([a a], [-50 50],'LineStyle',':','Color','r');
line([b b], [-50 50],'LineStyle',':','Color','r');
line([c c], [-50 50],'LineStyle',':','Color','r');
line([d d], [-50 50],'LineStyle',':','Color','r');
title('My')

figure(6);
plot(t,Mz,'b')
hold on;
line([3800 3800], [-50 50],'LineStyle',':','Color','r');
line([5800 5800], [-50 50],'LineStyle',':','Color','r');
line([a a], [-50 50],'LineStyle',':','Color','r');
line([b b], [-50 50],'LineStyle',':','Color','r');
line([c c], [-50 50],'LineStyle',':','Color','r');
line([d d], [-50 50],'LineStyle',':','Color','r');
title('Mz')

% Calculating mean values in the defined intervals
Fxfirstline = mean(F(a:b,1));
Fyfirstline = mean(F(a:b,2));
Fzfirstline = mean(F(a:b,3));
Mxfirstline = mean(F(a:b,4));
Myfirstline = mean(F(a:b,5));
Mzfirstline = mean(F(a:b,6));

Fxfirstturn = mean(F(b:c,1));
Fyfirstturn = mean(F(b:c,2));
Fzfirstturn = mean(F(b:c,3));
Mxfirstturn = mean(F(b:c,4));
Myfirstturn = mean(F(b:c,5));
Mzfirstturn = mean(F(b:c,6));

Fxsecondline = mean(F(c:d,1));
Fysecondline = mean(F(c:d,2));
Fzsecondline = mean(F(c:d,3));
Mxsecondline = mean(F(c:d,4));
Mysecondline = mean(F(c:d,5));
```

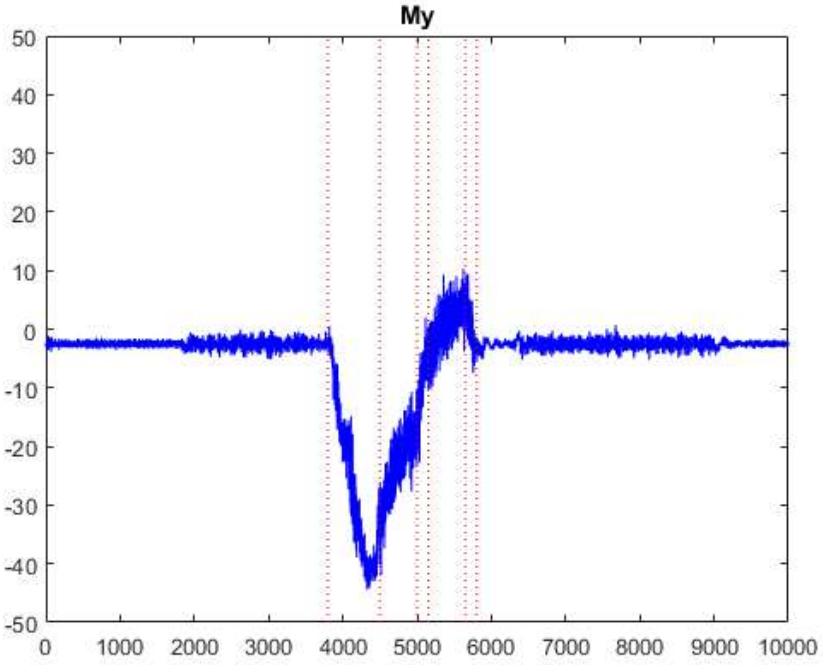
```
Mzsecondline = mean(F(c:d,6));

Fxsecondturn = mean(F(d:5800,1));
Fysecondturn = mean(F(d:5800,2));
Fzsecondturn = mean(F(d:5800,3));
Mxsecondturn = mean(F(d:5800,4));
Mysecondturn = mean(F(d:5800,5));
Mzsecondturn = mean(F(d:5800,6));
```

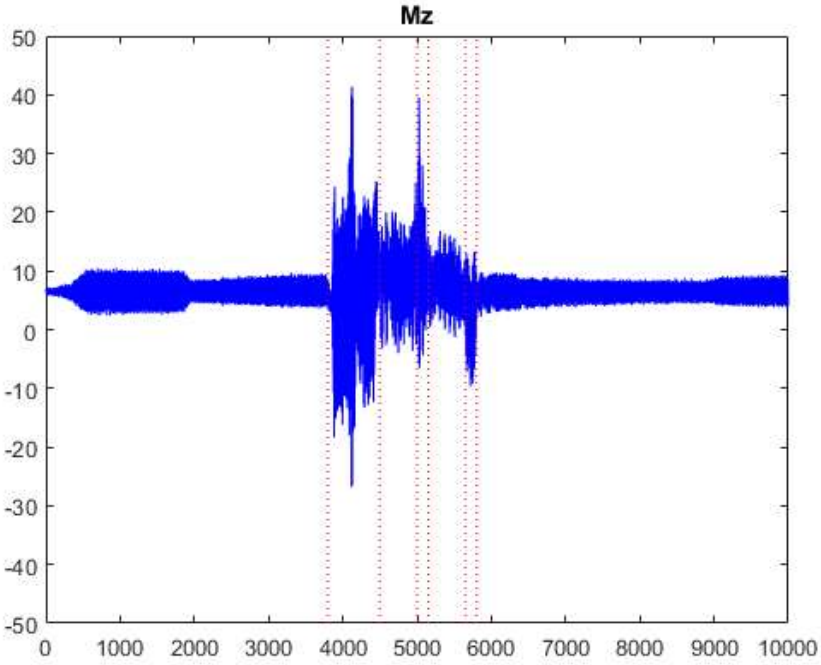



Fy
-160
-180
-200
-220
-240
-260
-280
-300
-320
-340
-360
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


Fz
150
100
50
0
-50
-100
-150
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


Mx
80
70
60
50
40
30
20
10
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

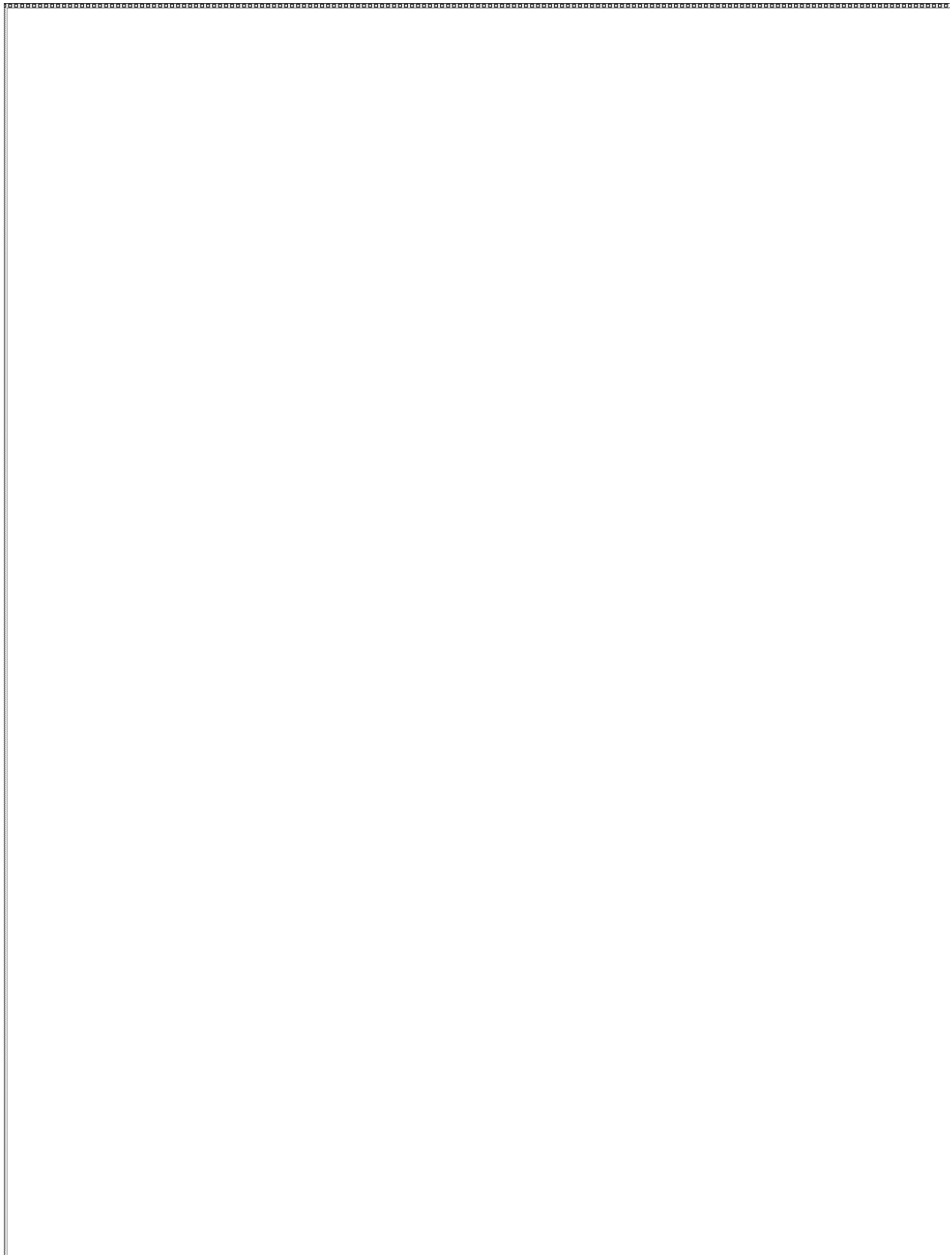

My
50
40
30
20
10
0
-10
-20
-30
-40
-50
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000



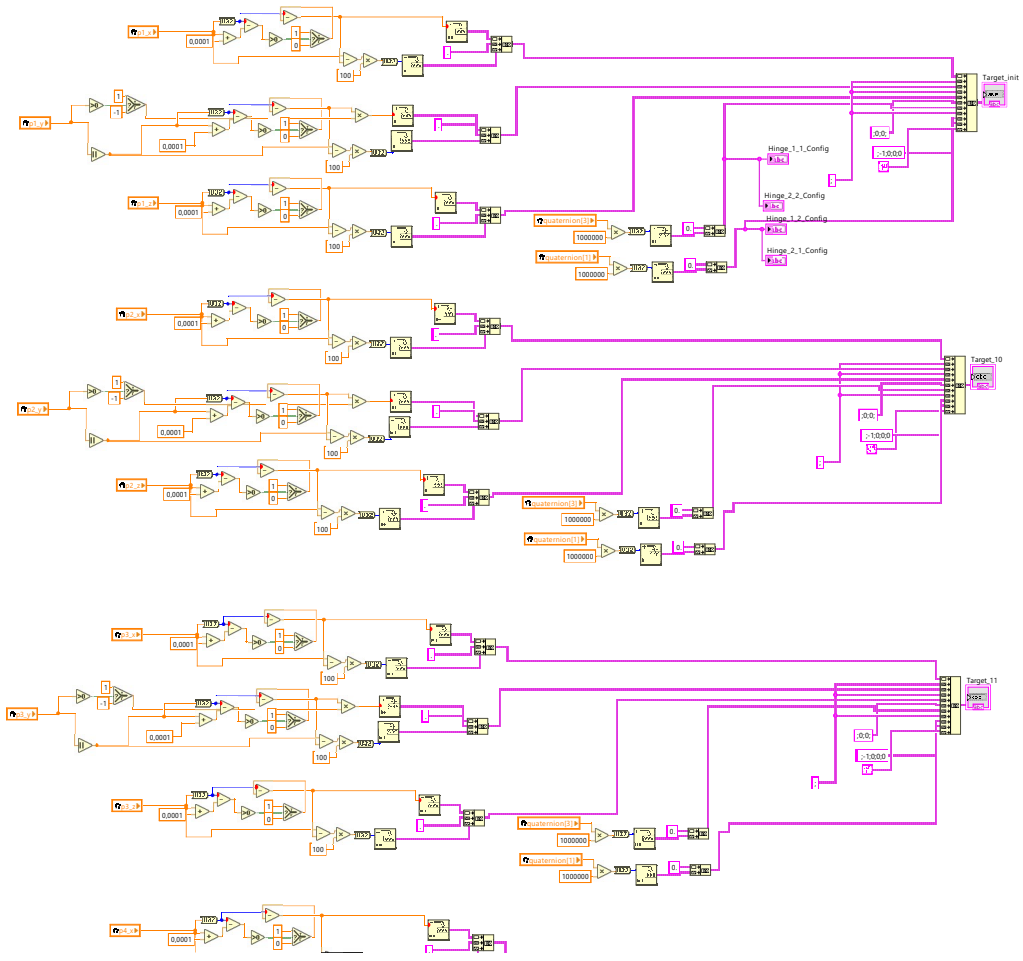
Published with MATLAB® R2017a

A.17 Labview Main Code

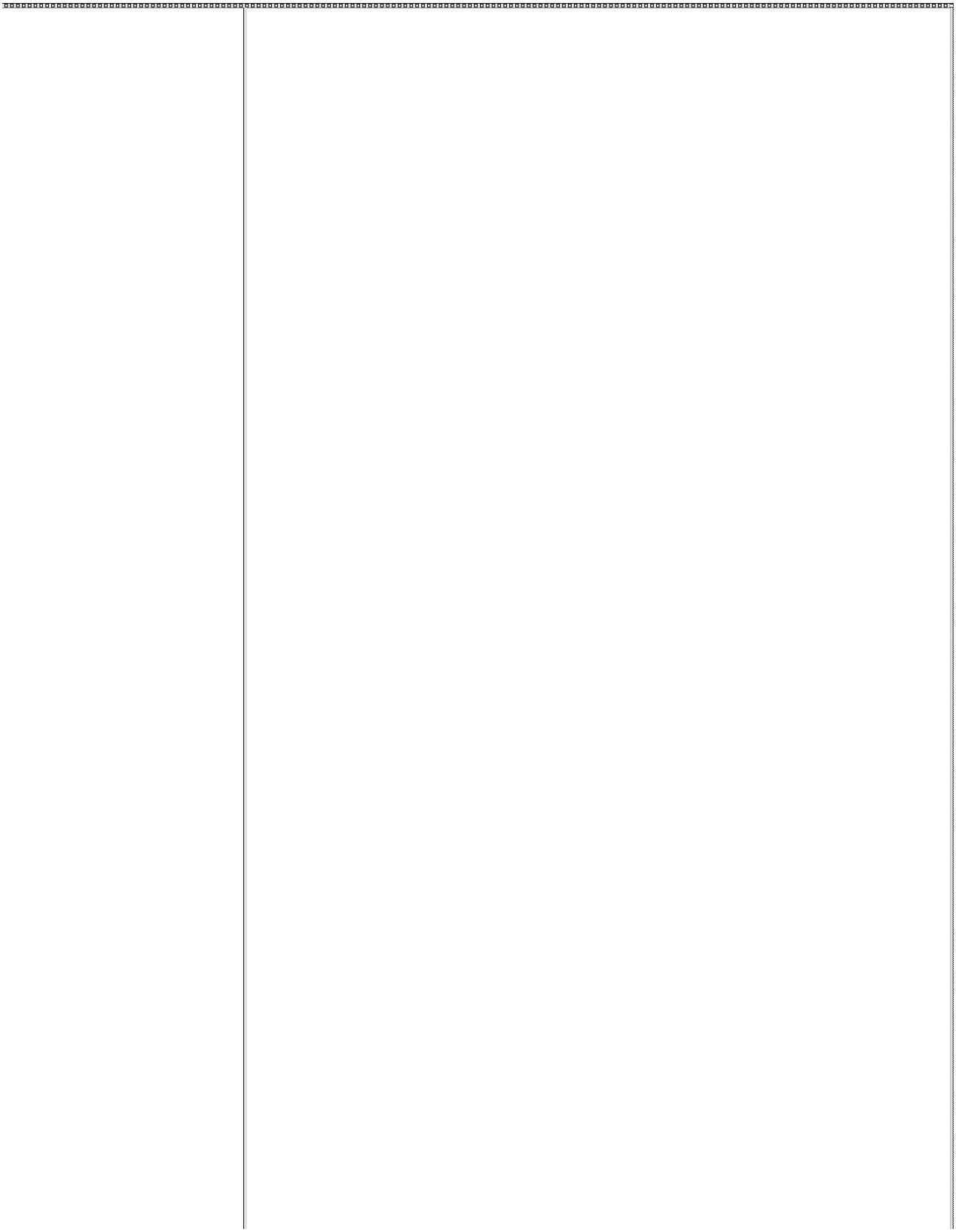
Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:42



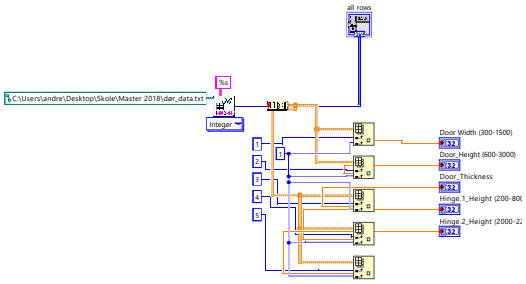
Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:43

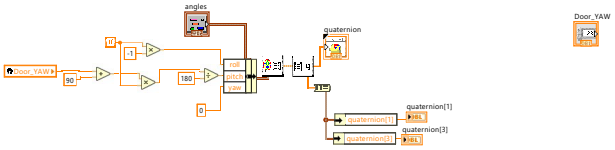


Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:43



Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:43

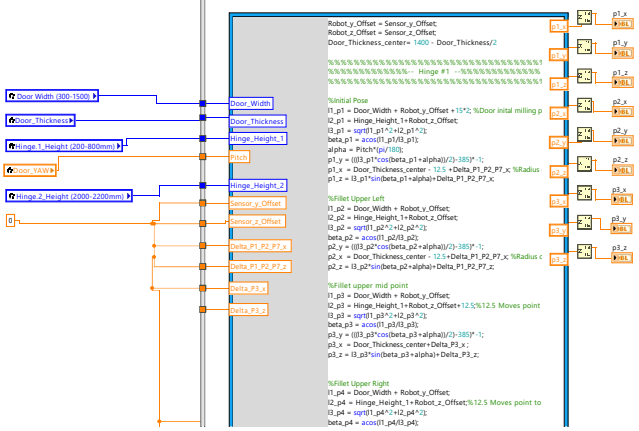




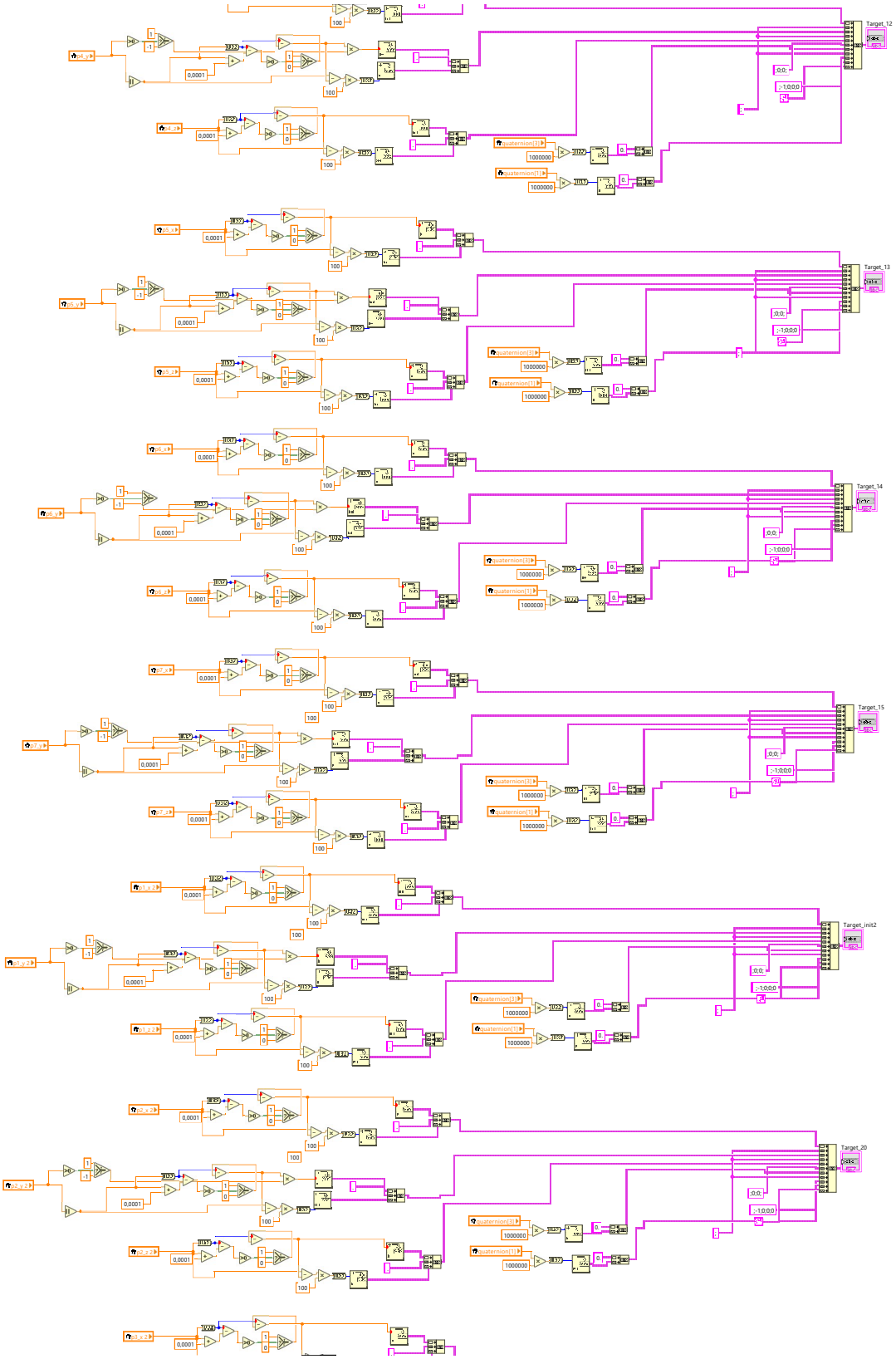
Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:43

0mm)

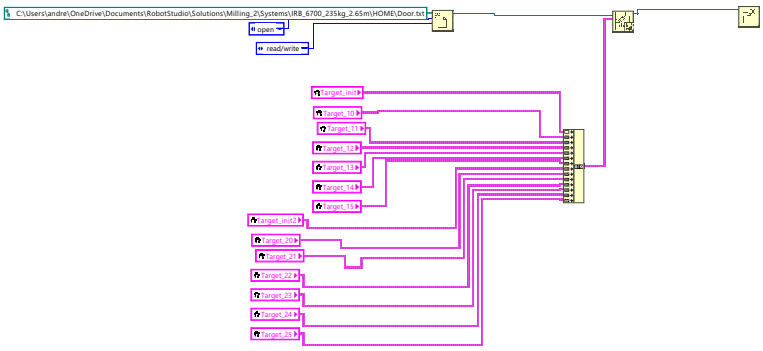
200mm)



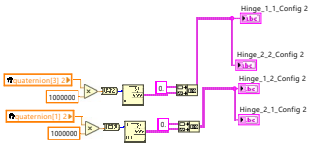
Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:43



Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:43

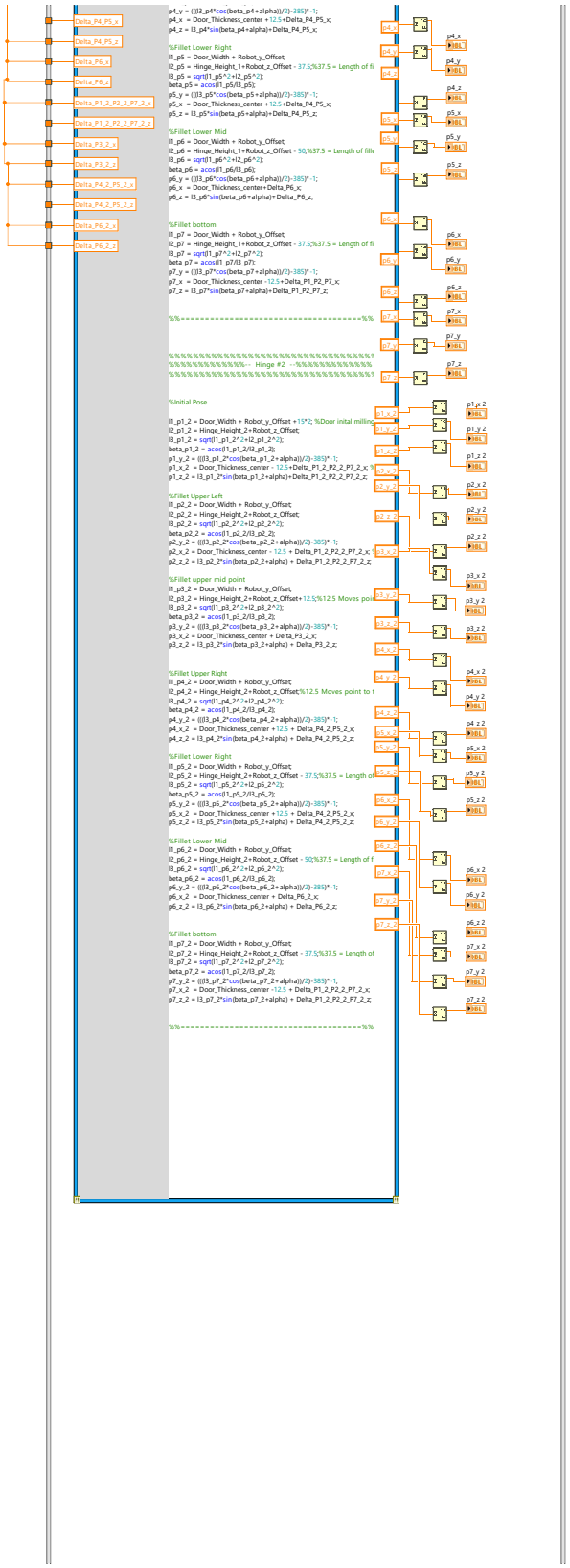


Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:44

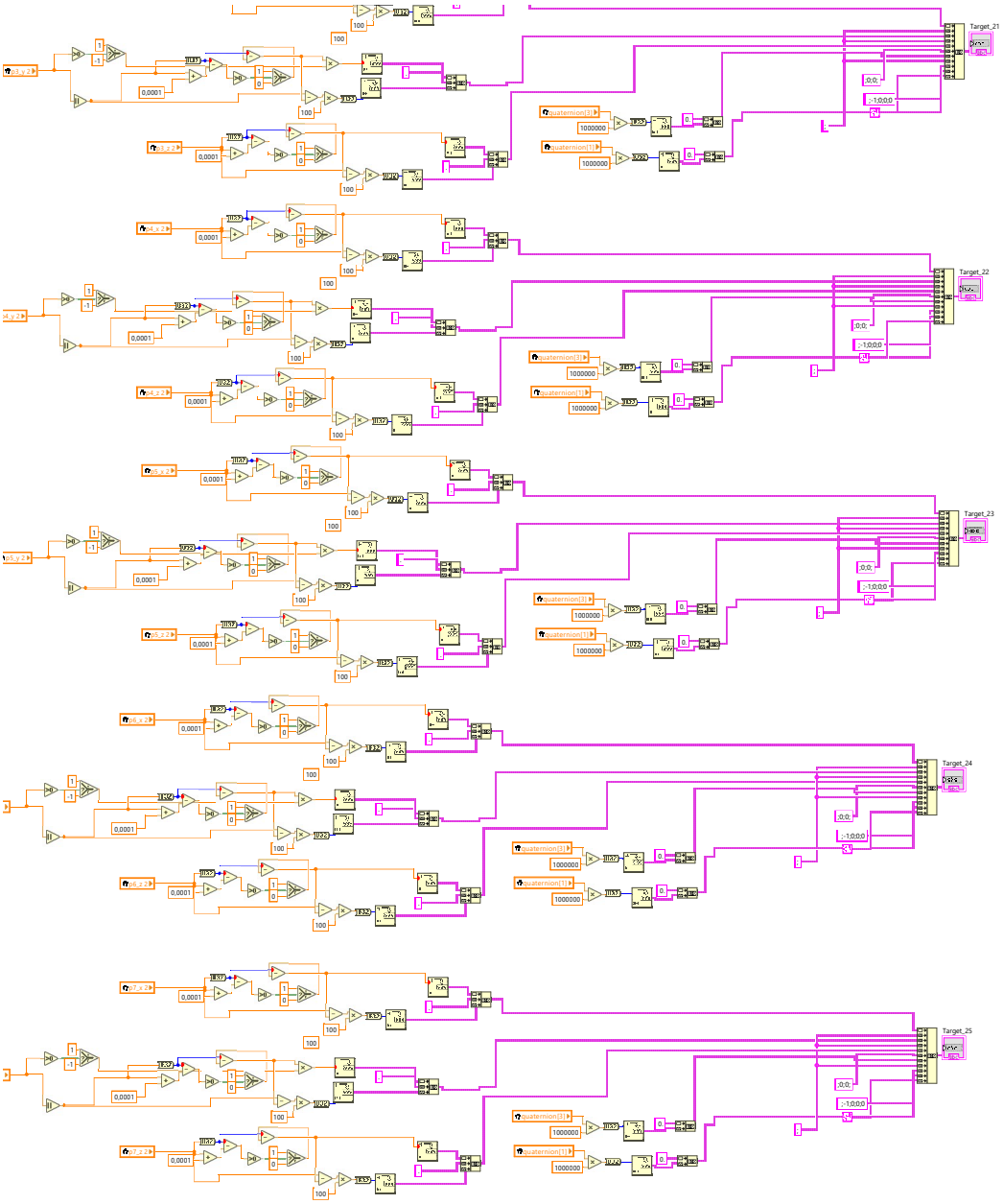


quaternion[1] 2
quaternion[3] 2

Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:44

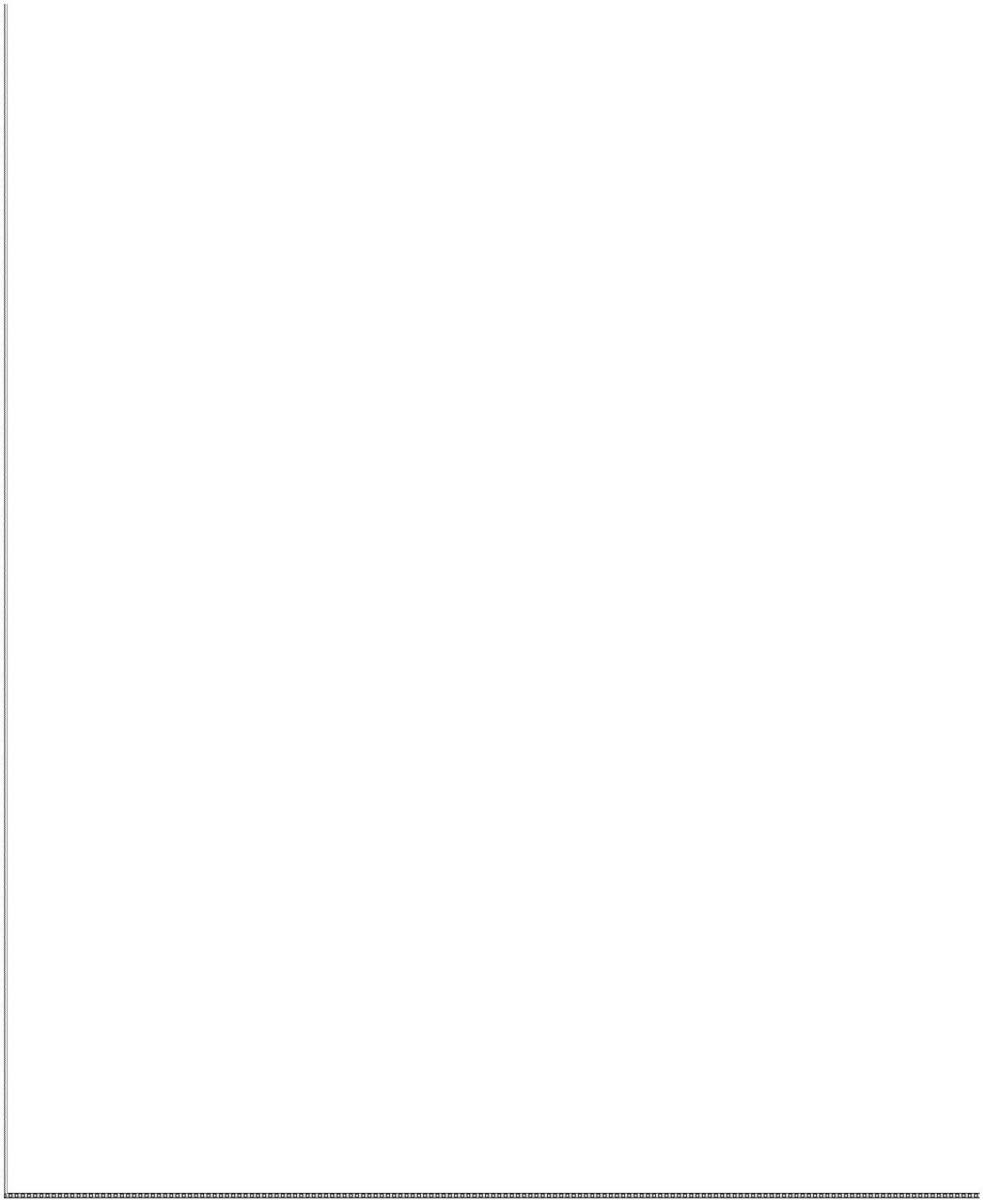


Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:44



Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:44

Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:44



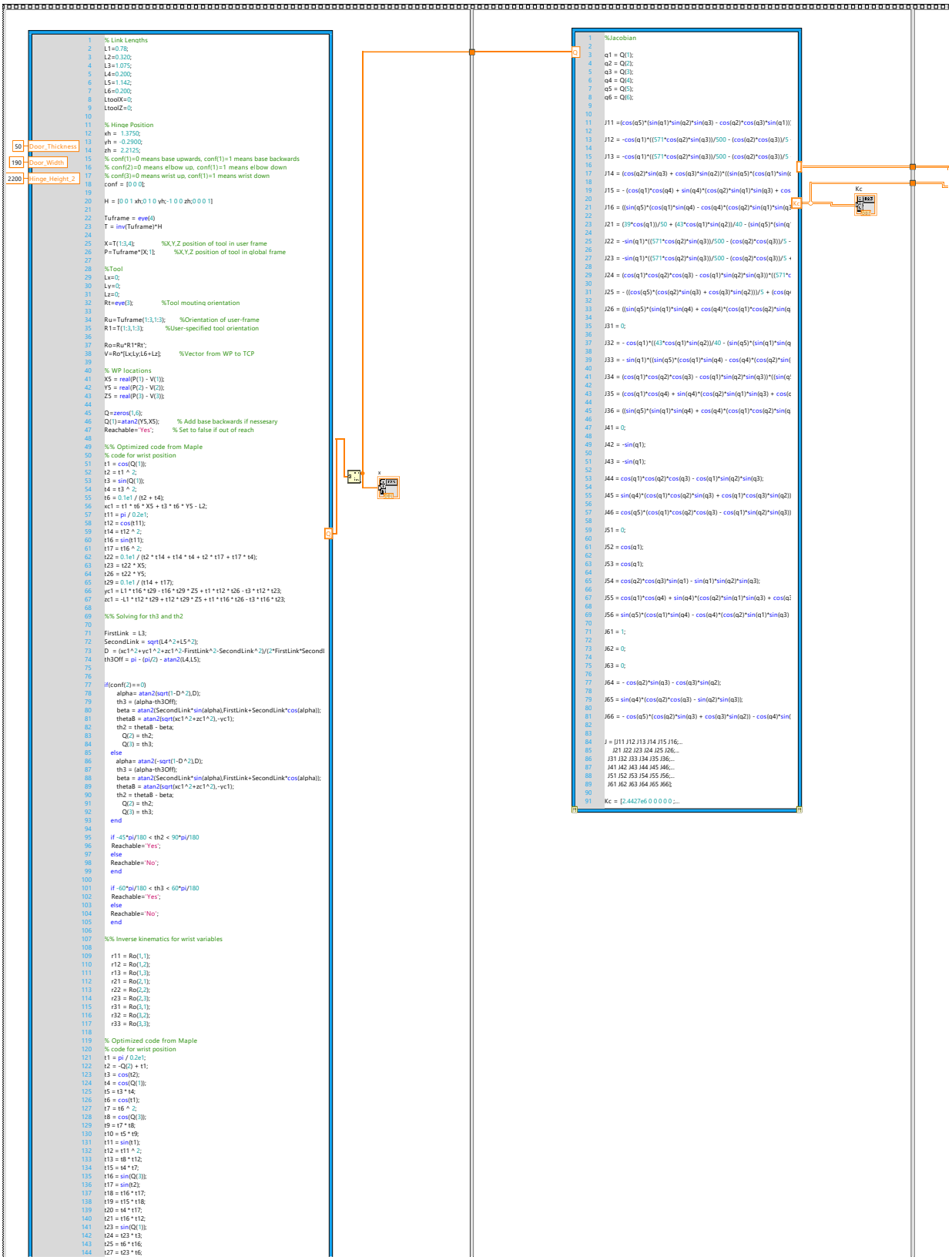
Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:45

Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:45

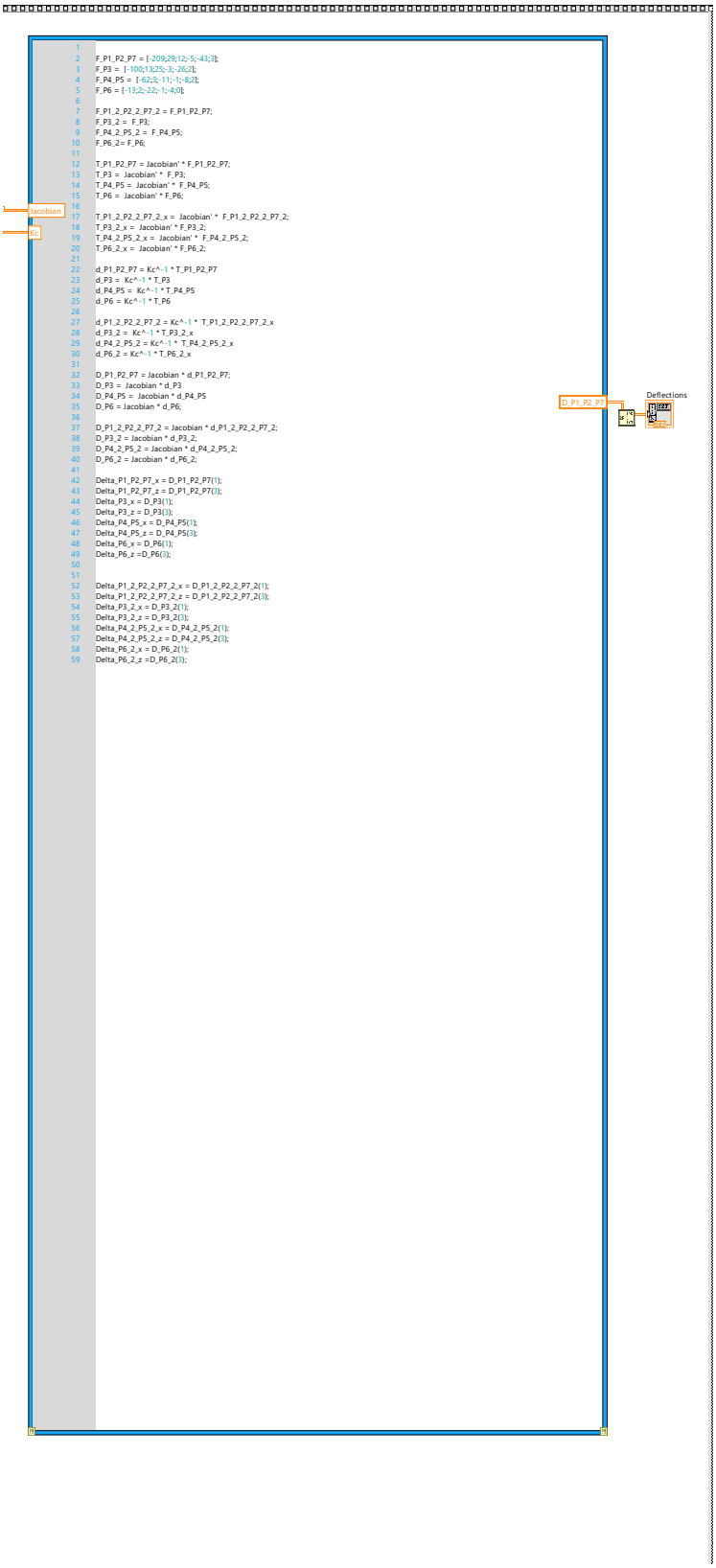
Write_Test_2.vi
C:\Users\andre\Desktop\Skole\Master 2018\Write_Test\Write_Test_2.vi
Last modified on 19.05.2018 at 10:43
Printed on 21.05.2018 at 09:45

A.18 Labview Stiffness Code

ProcessForce.vi
C:\Users\andre\Desktop\Skole\Master 2018\LabView_Vision\ProcessForce.vi
Last modified on 19.05.2018 at 08:21
Printed on 21.05.2018 at 09:40



ProcessForce.vi
C:\Users\andre\Desktop\Skole\Master 2018\LabView_Vision\ProcessForce.vi
Last modified on 19.05.2018 at 08:21
Printed on 21.05.2018 at 09:40




```
F_P1_P2_P7 = [-209;29;12;-5;-43;3];
F_P3 =  [-100;13;25;-3;-26;2];
F_P4_P5 = [-62;3;-11;-1;-8;2];
F_P6 = [-13;2;-22;-1;-4;0];

F_P1_2_P2_2_P7_2 = F_P1_P2_P7;
F_P3_2 =  F_P3;
F_P4_2_P5_2 =  F_P4_P5;
F_P6_2= F_P6;

T_P1_P2_P7 = Jacobian' * F_P1_P2_P7;
T_P3 =  Jacobian' *  F_P3;
T_P4_P5 =  Jacobian' *  F_P4_P5;
T_P6 =  Jacobian' * F_P6;

T_P1_2_P2_2_P7_2_x =  Jacobian' *  F_P1_2_P2_2_P7_2;
T_P3_2_x =  Jacobian' * F_P3_2;
T_P4_2_P5_2_x =  Jacobian' *  F_P4_2_P5_2;
T_P6_2_x =  Jacobian' * F_P6_2;

d_P1_P2_P7 = Kc^-1 * T_P1_P2_P7
d_P3 =  Kc^-1 * T_P3
d_P4_P5 =  Kc^-1 * T_P4_P5
d_P6 = Kc^-1 * T_P6

d_P1_2_P2_2_P7_2 = Kc^-1 *  T_P1_2_P2_2_P7_2_x
d_P3_2 =  Kc^-1 * T_P3_2_x
d_P4_2_P5_2 = Kc^-1 *  T_P4_2_P5_2_x
d_P6_2 = Kc^-1 * T_P6_2_x

D_P1_P2_P7 = Jacobian * d_P1_P2_P7;
D_P3 =  Jacobian * d_P3
D_P4_P5 =  Jacobian * d_P4_P5
D_P6 = Jacobian * d_P6;

D_P1_2_P2_2_P7_2 = Jacobian * d_P1_2_P2_2_P7_2;
D_P3_2 = Jacobian * d_P3_2;
D_P4_2_P5_2 = Jacobian * d_P4_2_P5_2;
D_P6_2 = Jacobian * d_P6_2;

Delta_P1_P2_P7_x = D_P1_P2_P7(1);
Delta_P1_P2_P7_z = D_P1_P2_P7(3);
Delta_P3_x = D_P3(1);
Delta_P3_z = D_P3(3);
Delta_P4_P5_x = D_P4_P5(1);
Delta_P4_P5_z = D_P4_P5(3);
Delta_P6_x = D_P6(1);
Delta_P6_z =D_P6(3);


Delta_P1_2_P2_2_P7_2_x = D_P1_2_P2_2_P7_2(1);
Delta_P1_2_P2_2_P7_2_z = D_P1_2_P2_2_P7_2(3);
Delta_P3_2_x = D_P3_2(1);
Delta_P3_2_z = D_P3_2(3);
Delta_P4_2_P5_2_x = D_P4_2_P5_2(1);
Delta_P4_2_P5_2_z = D_P4_2_P5_2(3);
Delta_P6_2_x = D_P6_2(1);
Delta_P6_2_z =D_P6_2(3);
```

ProcessForce.vi
C:\Users\andre\Desktop\Skole\Master 2018\LabView_Vision\ProcessForce.vi
Last modified on 19.05.2018 at 08:21
Printed on 21.05.2018 at 09:40

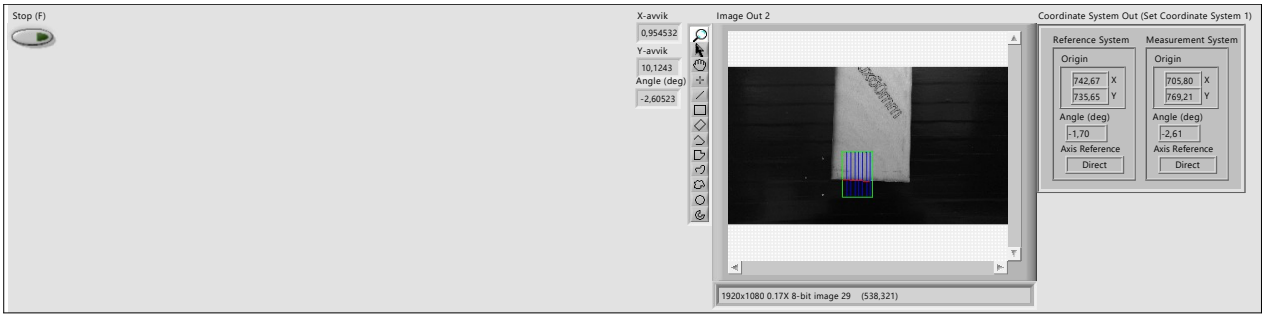
```
t28 = t8 * t17;
t31 = t23 ^ 2;
t32 = t3 ^ 2;
t33 = t31 * t32;
t34 = t16 ^ 2;
t35 = t7 * t34;
t37 = t8 ^ 2;
t38 = t7 * t37;
t40 = t34 * t12;
t42 = t37 * t12;
t44 = t31 * t7;
t45 = t17 ^ 2;
t46 = t34 * t45;
t48 = t37 * t45;
t50 = t31 * t45;
t53 = t4 ^ 2;
t54 = t32 * t53;
t59 = t53 * t7;
t62 = t53 * t45;
t65 = t33 * t35 + t33 * t38 + t33 * t40 + t33 * t42 + t54 * t35 + t54...
   * t38 + t50 * t40 + t54 * t40 + t62 * t40 + t50 * t42 + t54 * t42...
   + t62 * t42 + t44 * t46 + t44 * t48 + t59 * t46 + t59 * t48;
t66 = 0.1e1 / t65;
t69 = t24 * t9;
t71 = t23 * t7;
t72 = t71 * t18;
t73 = t23 * t17;
t76 = t4 * t6;
t84 = t32 * t7;
t85 = t84 * t34;
t86 = t84 * t37;
t87 = t32 * t34;
t89 = t32 * t37;
t91 = t7 * t45;
t92 = t91 * t34;
t93 = t91 * t37;
t94 = t46 * t12;
t95 = t48 * t12;
r = -(-t5 * t13 - t20 * t21 + t24 * t25 - t27 * t28 - t10 - t19) * t66...
   * r13 + (t24 * t13 + t73 * t21 + t5 * t25 - t76 * t28 + t69 + t72)...
   * t66 * r23 - (t16 * t3 - t28) * t11 / (t87 * t12 + t89 * t12 + t85...
   + t86 + t92 + t93 + t94 + t95) * r33;
t100 = t23 * t32;
t103 = t23 * t45;
t106 = t7 * t6;
t107 = t106 * t16;
t109 = t25 * t12;
t113 = t28 * t12;
t116 = t7 ^ 2;
t117 = t116 * t34;
t119 = t116 * t37;
t121 = t85 * t12;
t124 = t38 * t12;
t127 = t12 ^ 2;
t128 = t34 * t127;
t130 = t37 * t127;
t132 = t31 * t116;
t141 = t33 * t117 + t33 * t119 + 0.2e1 * t33 * t121 + 0.2e1 * t33 *...
   t124 + t33 * t128 + t50 * t128 + t33 * t130 + t50 * t130 + t132...
   * t46 + t132 * t48 + 0.2e1 * t44 * t94 + 0.2e1 * t44 * t95;
t150 = t53 * t116;
t159 = t54 * t117 + t54 * t119 + 0.2e1 * t54 * t121 + 0.2e1 * t54...
   * t124 + t54 * t128 + t62 * t128 + t54 * t130 + t62 * t130 + t150...
   * t46 + t150 * t48 + 0.2e1 * t59 * t94 + 0.2e1 * t59 * t95;
t161 = 0.1e1 / (t141 + t159);
t164 = t32 * t4;
t167 = t4 * t45;
t197 = t46 * t116 + t48 * t116 + t117 * t32 + t119 * t32 + t128 * t32...
   + t128 * t45 + t130 * t32 + t130 * t45 + 0.2e1 * t40 * t84 + 0.2e1...
   * t40 * t91 + 0.2e1 * t42 * t84 + 0.2e1 * t42 * t91;
t198 = 0.1e1 / t197;
s = (t4 * t106 * t28 + t100 * t40 + t100 * t42 + t103 * t40 + t103...
   * t42 - t5 * t107 - t5 * t109 + t76 * t113 - t69 - t72) * t161...
   * r13 - (-t23 * t106 * t28 + t24 * t107 + t24 * t109 - t27 * t113...
   + t164 * t40 + t164 * t42 + t167 * t40 + t167 * t42 - t10 - t19)...
   * t161 * r23 - (t8 * t3 + t18 + t46 + t48 + t87 + t89) * t6 * t11...
   * t198 * r33;
t202 = t6 * t34;
t204 = t6 * t37;
t208 = t7 * t16;
t213 = t6 * t8;
t217 = t11 * (t100 * t202 + t100 * t204 - t20 * t13 - t15 * t28 + t27...
   * t18 + t5 * t208 + t5 * t21 + t24 * t213 + t27 * t46 + t27 * t48);
t231 = t11 * (-t73 * t13 - t164 * t202 - t164 * t204 - t76 * t18 + t24...
   * t208 + t24 * t21 - t5 * t213 - t71 * t28 - t76 * t46 - t76 * t48);
t239 = (-t12 * t17 * t16 - t12 * t3 * t8 + t85 + t86 + t92 + t93) * t198;
t = -t217 * t161 * r11 - t231 * t161 * r21 + t239 * r31;
u = -t217 * t161 * r12 - t231 * t161 * r22 + t239 * r32;
v = -t217 * t161 * r13 - t231 * t161 * r23 + t239 * r33;

TA = zeros(3,3);

TA(1,3) = r;
TA(2,3) = s;
TA(3,1) = t;
TA(3,2) = u;
TA(3,3) = v;

%% Solving for th5, th4 and th6

    q6a=acos(TA(3,3));
q6b=acos(TA(3,3))+pi/2;
if(conf(3)==0) %Wrist up
Q(4)=atan2(-TA(2,3),-TA(1,3));
Q(5)=min(q6a,q6b);
Q(6)=atan2(TA(3,2),-TA(3,1));
 else      %Wrist down
Q(4)=atan2(TA(2,3),TA(1,3));
Q(5)=-max(q6a,q6b)+(pi/2);
Q(6)=atan2(-TA(3,2),TA(3,1));
 end

 if -90*pi/180 < Q(5) < 90*pi/180
  Reachable='Yes';
 else
 Reachable='No';
 end
```

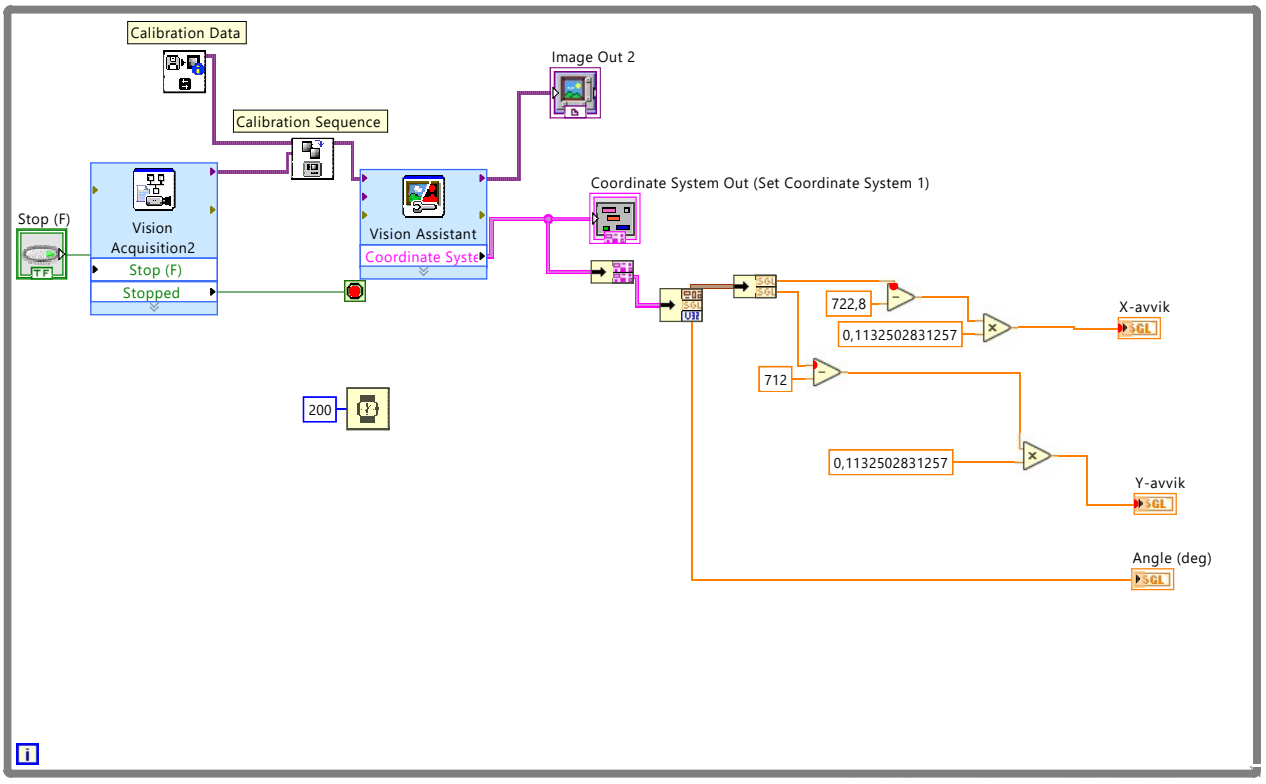
ProcessForce.vi
C:\Users\andre\Desktop\Skole\Master 2018\LabView_Vision\ProcessForce.vi
Last modified on 19.05.2018 at 08:21
Printed on 21.05.2018 at 09:40

A.19 Labview Camera Code

Vision_Test_3.vi
C:\Users\andre\Desktop\Skole\Master 2018\LabView_Vision\Vision_Test_3.vi
Last modified on 19.05.2018 at 08:05
Printed on 22.05.2018 at 10:49

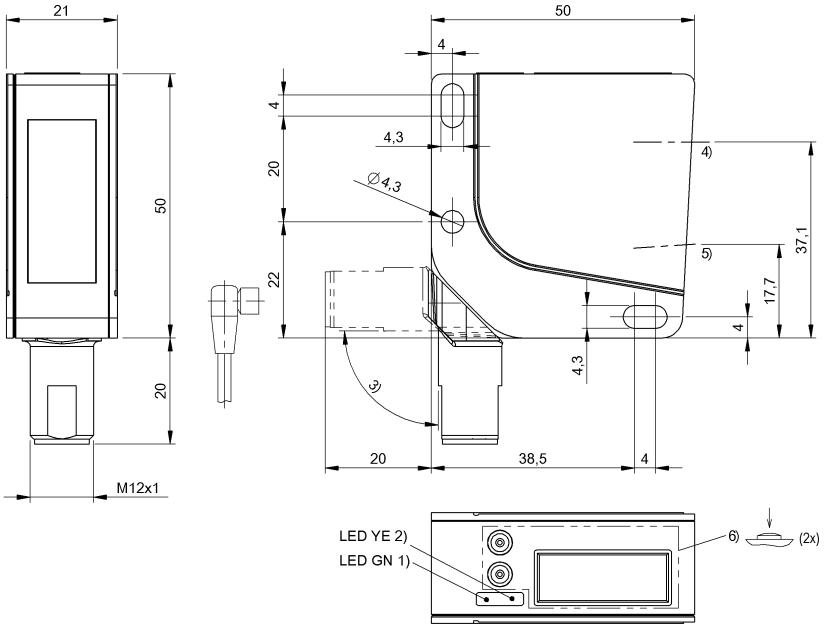
Vision_Test_3.vi





A.20 Data Balluff BOD0025

Photoelectric Sensors
BOD 24K-LB03-S92
Ordercode: BOD0025



1) Operating voltage 2) Output function 3) rotatable 180° 4) Optical axis emitter 5) Optical axis receiver 6) Display and keypad

Display/Operation

Adjuster	Key (2x)
Display	Error - LED green, flashing Teach-in - LED yellow/green, flashing LED green: Power Object in range - LED yellow
Power indicator	yes
Setting	Switching output PNP/NPN Normally open/Normally closed Light-on/dark-on Teaching switchpoints Factory setting (Reset) calibration mode Working range

Electrical connection

Connection	Connector, M12x1-Male, 5-pole
Polarity reversal protected	yes
Short-circuit protection	yes

Electrical data

Load resistance RL max. (Analog I)	500 Ohm
No-load current Io max. at Ue	180 mA
Operating voltage Ub	18…30 VDC
Protection class	II
Rated operating current Ie	100 mA
Rated operating voltage Ue DC	24 V
Ready delay tv max.	300 ms
Residual ripple max. (% of Ue)	15 %
Switching frequency	500 Hz
Turn-off delay toff max.	5 ms
Turn-on delay ton max.	5 ms

Environmental conditions

Ambient temperature	-20…50 °C
IP rating	IP67

Functional safety

MTTF (40 °C)	37 a

General data

Application	Distance measurement
Approval/Conformity	CE cULus EAC

Internet	www.balluff.com
Balluff Germany	+49 (0) 7158 173-0, 173-370
Balluff USA	1-800-543-8390
Balluff China	+86 (0) 21-50 644131

For definitions of terms, see main catalog
Subject to change without notice [252808]

eCl@ss 9.1: 27-27-08-01
ETIM 6.0: EC001825
BOD0025_0.25_2018-05-04

1(3)

Photoelectric Sensors

BOD 24K-LB03-S92
Ordercode: BOD0025

Basic standard	IEC 60947-5-2
Principle of operation	Photoelectric Distance Sensor BOD
Series	24K
Style	Square Connection can be rotated

Material

Housing material	Plastic
Material sensing surface	Glass

Mechanical data

Dimension	50 x 21 x 50 mm
Fastening detail	Screw M4

Optical data

Ambient light max.	5000 Lux
Beam characteristic	Divergent
Laser class per IEC 60825-1	2
Light spot size	1 x 1 mm at 450 mm
Light type	Laser red light
Principle of optical operation	Triangulation
Pulse duration t max.	22 ms
Pulse power Pp max.	1.2 mW
Switching function optical	Light/dark switching
Wave length	655 nm

Output/Interface

Analog output	Analog, current 4…20 mA
Switching output	2x PNP/NPN NO/NC push-pull

Range/Distance

Accuracy	±1 %FS
Range	50…650 mm
Rated operating distance Sn	650 mm Adjustable
Repeat accuracy	0.5 % FS
Resolution	≤ 100 µm

Remarks

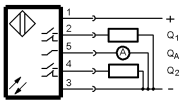
For additional information, refer to user's guide.

For further information about the MTTF and B10d see MTTF / B10d certificate

Indication of the MTTF- / B10d value does not represent a binding composition and/or life expectancy assurance; these are simply experiential values with no warranty implications. These declared values also do not extend the expiration period for defect claims or affect it in any way.

Connector view

Wiring Diagram



Symbols for Optoelectronic Sensors

Warning Symbols

LASER BEAM - DO NOT STARE INTO THE LIGHT BEAM!

LASER CLASS 2 per IEC60825-1: 2003-10

A.21 Data OPT Short Range

For the latest prices, please check AutomationDirect.com.

OPT Short Range (CMOS) Series Photoelectric Sensors

OPT2001

50 x 50 mm rectangular plastic - DC

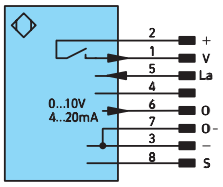
- Diffuse (Reflex) laser distance measurement sensors with CMOS technology
- Analog and switching outputs available
- Measured value independent of material, color, and brightness
- Class 1 and 2 lasers available (safety label included with Class 2 lasers)
- High resolution down to 8 µm - (analog scalable down to 5 mm range)
- High speed response times down to 660 µs
- M12 quick-disconnect; order cable separately
- Mounting hardware included

OPT Series Photoelectric Sensors Selection Chart

Part Number	Price	Sensing Range	Laser Class	Measurement Rate	Resolution	Output State	Logic	Connection	Wiring	Characteristic Curves
Diffuse (Reflex)										
OPT2001	$629.00	30-80 mm [1.18 - 3.15 in]	2	1500/s (660 µs)	<8 µm	Analog 4-20 mA or 0-10 V	—	8-pin M12 quick-disconnect	Diagram 1	See Characteristic Curve
OPT2002	$629.00		1	1000/s (1000 µs)			—			
OPT2003	$629.00	40-160 mm [1.57 - 6.30 in]	2	1500/s (660 µs)	<20 µm		—			
OPT2004	$629.00		1	1000/s (1000 µs)			—			
OPT2005	$629.00	50-350 mm [1.97 - 13.80 in]	2	800/s (1250 µs)	<50 µm		—			
OPT2006	$629.00		1	500/s (2000 µs)			—			
OPT2007	$319.00	0 - 660 mm [0 - 25.98 in] working range 60-660 mm [2.36 - 25.98 in] adjustable range	1	100 Hz switching	Hysteresis <1 % of range	Selectable (N.O., N.C.)	5-wire, configurable as PNP, NPN, or Push-Pull	5-pin M12 quick-disconnect	Diagram 2	—

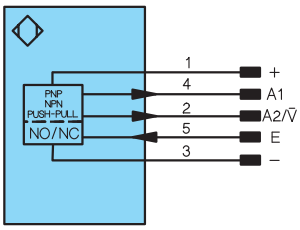
Wiring Diagrams

Diagram 1



+	Supply Voltage "+"
V	Contamination/Error output (NO)
O	Analog output
O–	Ground for the analog output
–	Supply Voltage "0 V"
S	Shielding
La	Emitted Light disengageable

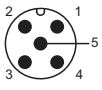
Diagram 2



+	Supply Voltage "+"
–	Supply Voltage "0 V"
A1/A2	Switching output (NO)
V̄	Contamination Warning/ Error Output (NC)
E	Input (Teach Input, Emitted light can be switched off)

Connectors

5-Pin M12 connector



8-Pin M12 connector

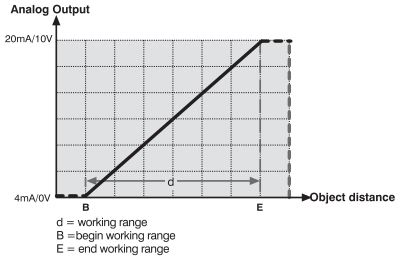
NOTE: CLASS 2 POWER SOURCE REQUIRED

PRODUCT MANUAL AVAILABLE VIA DOWNLOAD AT WWW.AUTOMATIONDIRECT.COM

OPT Short Range (CMOS) Series Photoelectric Sensors

Specifications	OPT 2001	OPT 2002	OPT 2003	OPT 2004	OPT 2005	OPT 2006	OPT 2007
Type	Diffuse Reflex						
Sensing Distance	30-80 mm [1.18-3.15 in]	30-80 mm [1.18-3.15 in]	40-160 mm [1.57- 6.30 in]	40-160 mm [1.57- 6.30 in]	50-350 mm [1.97-13.78 in]	50-350 mm [1.97-13.78 in]	60-660 mm [2.36-25.98 in]]
Light Spot Diameter (at maximum range)	1 x 2 mm [0.04 x 0.08 in]	0.7 x 1.4 mm [0.03 x 0.06 in]	1 x 2.5 mm [0.04 x 0.10 in]	0.9 x 1.8 mm [0.04 x 0.07 in]	1.5 x 4 mm [0.06 x 0.16 in]	1.4 x 3.1 mm [0.06 x 0.12 in]	2.0 x 5.5 mm [0.08 x 0.22 in]
Emission	Class 2 Red laser 660 Nm	Class 1 Red laser 660 Nm	Class 2 Red laser 660 Nm	Class 1 Red laser 660 Nm	Class 2 Red laser 660 Nm	Class 1 Red laser 660 Nm	Class 1 Red laser 655 Nm
Sensitivity	Adjustable via Teach						
Output Type	0-10 VDC or 4-20 mA: PNP error output						Complementary N.O./N.C. (Light-on, Dark-on) PNP or NPN
Current Output Max Load	500Ω						NA
Voltage Output Min Load	10 KΩ						NA
Operating Voltage	18-30 VDC						10-30 VDC
No Load Supply Current	<80 mA @ 24 VDC						<50 mA @24 VDC
Operating (Load) Current	max 200 mA						
Off-state (Leakage) Current	negligible						
Voltage Drop	<2.5V						<1.5V
Measurement Rate/ Resolution	1500/s (660 µs) @ 12µm 600/s(1660 µs) @ 8µm	1000/s (1000 µs) @ 12 µm 500/s (2000 µs) @ 8 µm	1500/s (660 µs) @ 30 µm 600/s (1660 µs) @ 20 µm	1000/s (1000 µs) @ 30 µm 500/s (2000 µs) @ 20 µm	800/s (1250 µs) @ 80 µm 400/s (2500 µs) @ 50 µm	500/s (2000 µs) @ 80 µm 250/s (4000 µs) @ 50 µm	NA
Switching Frequency	1.5 kHz	1.0 kHz	1.5 kHz	1.0 kHz	800 Hz	500 Hz	100 Hz
Linearity	0.1%				0.15%		NA
Time Delay Before Availability (tv)	NA						
Short-Circuit Protection	Yes						
Operating Temperature	-25°C to 50°C [13°F to 122°F]						-25°C to 60°C [13°F to 140°F]
Protection Degree (DIN 40050)	IEC IP67						IEC IP68
LED Indicators - Switching Status	Yellow						
LED Indicators - Power	Green						
Housing Material	Polycarbonate						
Lens Material	Poly(methyl methacrylate) (PMMA)						
Shock/Vibration	See *Terminology* section.						
Tightening Torque	0.5 N·m (mounting screws)						
Weight (lbs) (cable/connector)	0.2						
Connectors	M12 Quick Disconnect						
Agency Approvals	CE, cULUS, E189727, RoHs						

Characteristic Curves



d = working range
B =begin working range
E = end working range

The Laser Classification Systems for the standards IEC (EN) 60825-1 defines the following safety classes:

Class 1

This class is eye-safe under all operating conditions.

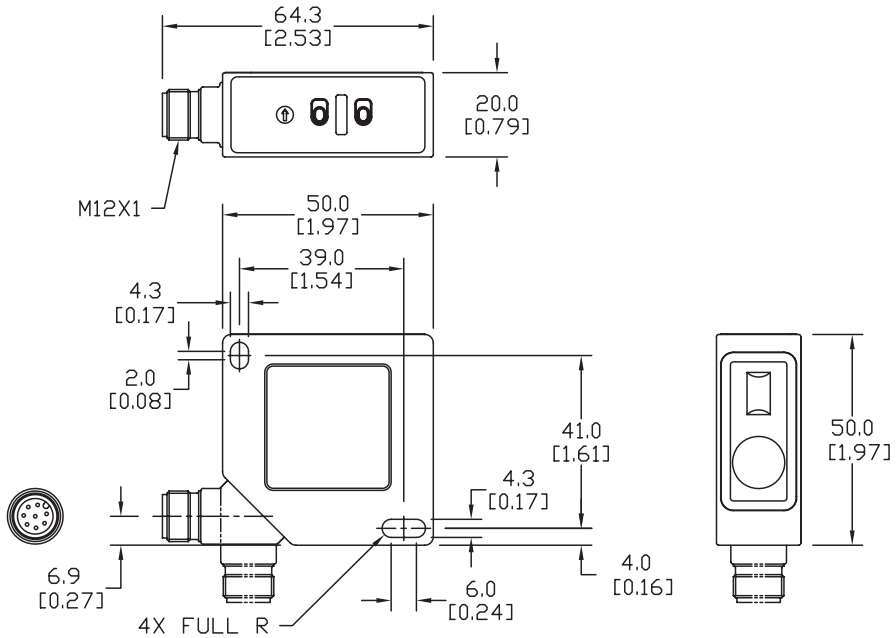
Class 2

These are visible lasers. This class is safe for accidental viewing under all operating conditions. However, it may not be safe for a person who deliberately stares into the laser beam for longer than 0.25 s, by overcoming their natural aversion response to the very bright light.

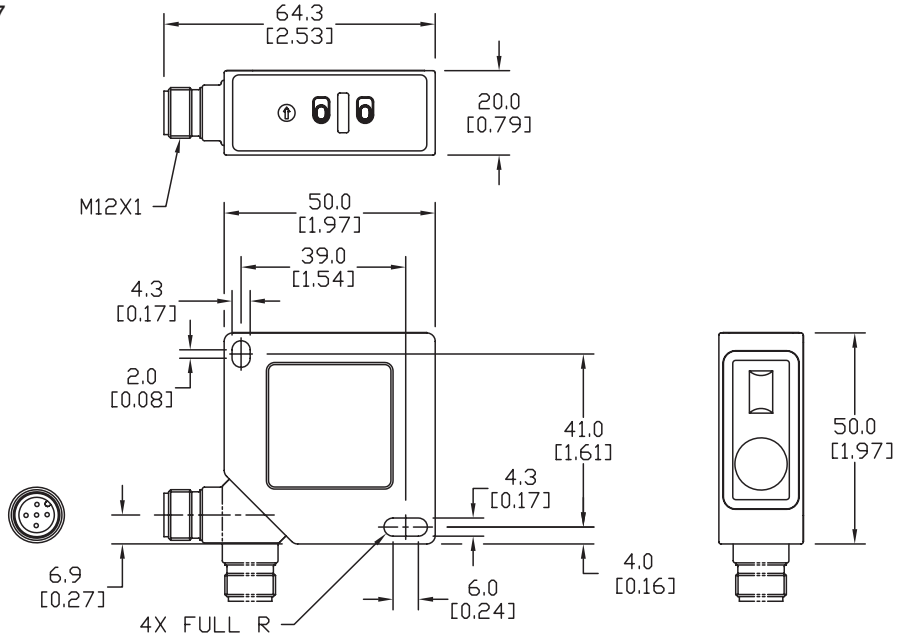
OPT Short Range (CMOS) Series Photoelectric Sensors

Dimensions mm [inches]

OPT2001
OPT2002
OPT2003
OPT2004
OPT2005
OPT2006



OPT2007



Accessories for OPT Series 50x50mm Photoelectric Sensors

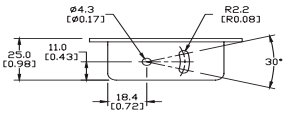
Right-angle Brackets

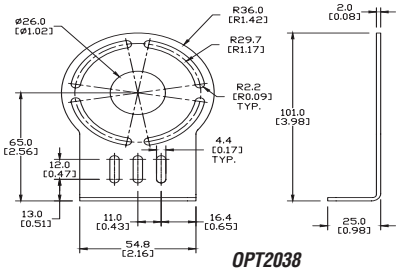
Mounting bracket, right-angle, nickel-plated steel or 304 stainless steel.
For use with OPT series 50x50mm photoelectric sensors.

Accessories for OPT2001 - OPT2011 Sensors			
Part Number	***Price***	***Description***	***Weight [lb]***
OPT2031	$5.50	Mounting bracket, right-angle, nickel-plated steel. For use with OPT series 50x50mm photoelectric sensors.	0.24
OPT2038	$8.50	Mounting bracket, right-angle, vertical and horizontal adjustment, 304 stainless steel. For use with OPT series 50x50mm photoelectric sensors.	0.19

Dimensions

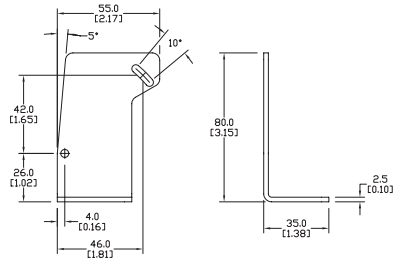
mm [inches]

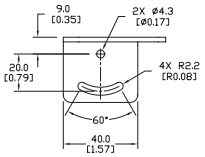




OPT2038

OPT2038





OPT2031

OPT2031

See our website: www.AutomationDirect.com for complete engineering drawings

Accessories for OPT Series 50x50mm Photoelectric Sensors

Right-angle Swivel Mounting Systems

Mounting bracket, right-angle swivel, 360 degree vertical and horizontal adjustment, 12mm rod mount. For use with OPT series 50x50mm photoelectric sensors. Available in all stainless steel or with an aluminum head with a stainless steel mounting plate.

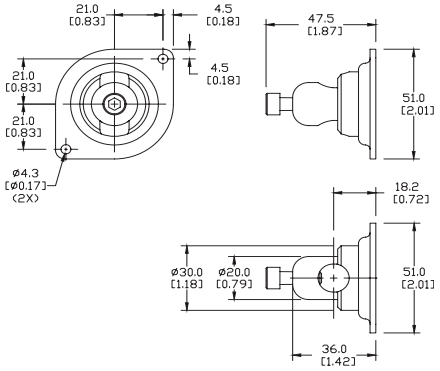
Accessories for OPT2001 - OPT2011 Sensors					
Part Number	***Price***	***Description***	***Mounting Head***	***Mounting Plate***	***Weight [lb]***
OPT2122	$9.00	Mounting bracket, right-angle swivel, 360 vertical and horizontal adjustment, aluminum, 12mm rod mount. For use with OPT series 50x50mm photoelectric sensors.	Aluminum	304 Stainless Steel	0.19
OPT2123	$16.00	Mounting bracket, right-angle swivel, 360 vertical and horizontal adjustment, stainless steel, 12mm rod mount. For use with OPT series 50x50mm photoelectric sensors.	304 Stainless Steel	304 Stainless Steel	0.31

Note: 304 Stainless steel mounting rods sold separately: OPT2109 (200mm length), OPT2110 (300mm length), and OPT2111 (500mm length).

OPT2122, OPT2123

Dimensions

mm [inches]



See our website: www.AutomationDirect.com for complete engineering drawings

A.22 Data Zivid

ZiVID

Zivid is the world's most accurate real-time 3D color camera

DEFINING THE FUTURE OF 3D MACHINE VISION

3D IMAGING WITHOUT COMPROMISES

Excellent data quality in HD at 10Hz with depth resolution of 0.1 mm. No compromises between speed and resolution.

FULL COLOR

Point cloud data (XYZ) and vivid RGB colors captured with the same sensor chip. One-to-one correspondence between color and depth.

HANDLES ANY MATERIAL

Captures high quality 3D images of even the most problematic industrial objects. Shiny metallic, black and absorbing or partly translucent.

EASE OF USE

Factory calibrated, easy and intuitive to use. Comes with interfaces for machine vision software and APIs for all major programming languages.

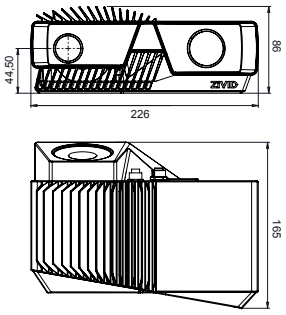
zividlabs.com

0.1mm	10Hz	3D + RGB	HDR
RESOLUTION	ACQUISITION RATE	OUTPUT	IMAGING

OUTPUT	2.3 Mpixel 3D RGBD image (X,Y,Z and R,G,B for each pixel)
ACQUISITION RATE	≥10 Hz 100ms snapshots
FIELD OF VIEW	780 x 490 mm @ 1.1m 425 x 267 mm @ 0.6m
OPTIMAL WORKING DISTANCE	0.6 - 1.1m
DEPTH RESOLUTION	0.1mm @ 0.6m
SOFTWARE APIS	C++, C#, .NET, Python, MATLAB
OS	Windows 7 / 8 / 10

HOUSING	Rugged aluminum Dust & water resistant
DIMENSION	226 x 165 x 86 mm
WEIGHT	2 kg
INPUT VOLTAGE	24VDC
INTERFACES & CONNECTORS	USB 3.0: Data I/O M12-5: Power + Status M12-8: Sync In + Sync Out
AVAILABLE USB 3.0 CABLES	3m, 5m, 10m, 25m
PC REQUIREMENTS	DirectX 11 compatible graphics card

APPLICATION AREAS
Object recognition & classification
Pick & place
In-line quality control
Kitting and robotic assembly operations
Automatic processing of food products
Research & development



zividlabs.com

Zivid Labs AS · info@zividlabs.com
Gaustadalléen 21 · 0349 Oslo · Norway

A.23 Chip Load Chart

The information given should only be used as a guideline or starting point for feed rate selection. Your actual feeds and speeds will vary widely as a result of "contributing factors" such as machine rigidity, horsepower, collet condition, spindle integrity, part clamping, hold down and many other factors. Generally speaking, solid carbide spiral tooling will perform better (i.e. longer life, less tool breakage) at faster feed rates. We recommend selecting a "starting point" feed rate and increasing that feed rate until part finish becomes undesirable or other limiting factors become evident.

Chip Load

The chip load is a measurement of the thickness of material removed by each cutting edge during a cut. This is a valuable piece of information which can then be used to calculate new set ups.

Calculations are as follows: Chip Load = Feed Rate (inches per minute) / (RPM x number of flutes)
Example: Chip Load = 500 inches per minute / (15,000 RPM x 2 flutes) Chip Load = .017"

Chip loads are based on material thickness of average size for cutting edge length of tool. These recommendations do not apply to thicker material or tools with long cutting edge lengths. These chiploads are only a recommended starting point and may not accomondate all circumstances. Therefore, tooling damage may still occur and use of this chart does not warranty against tool breakage.

We would strongly encourage you to consult us directly on new tool applications. Our staff would be happy to discuss any technical questions by phone or email.

Chip Load Chart

Tool Diameter	Hard Wood	Softwood Plywood	MDF/Particle Board	High Pressure Laminate **	Phenolic **
1/8"	.003" - .005"	.004" - .006"	.004" - .007"	.003" - .005"	.004" - .005"
1/4"	.009" - .011"	.011" - .013"	.013" - .016"	.009" - .012"	.011" - .012"
3/8"	.015" - .018"	.017" - .020"	.020" - .023"	.015" - .018"	.017" - .018"
1/2" & up	.019" - .021"	.021" - .023"	.025" - .027"	.023" - .025"	.024" - .026"

**Recommended RPM 9-10,000

Tool Diameter	Hard Plastic	Soft Plastic	Solid Surface	Acrylic	Aluminum
1/8"	.002" - .004"	.003" - .006"	.002" - .004"	.003" - .005"	.003" - .004"
1/4"	.006" - .009"	.007" - .010"	.006" - .009"	.008" - .010"	.005" - .007"
3/8"	.008" - .010"	.010" - .012"	.008" - .010"	.010" - .012"	.006" - .008"
1/2" & up	.010" - .012"	.012" - .016"	.010" - .012"	.012" - .015"	.008" - .010"

Other Valuable Formulas:

Feed Rate = RPM x number of flutes x chip load
RPM = feed rate/(number of flutes x chipload)
Metric Conversion: Divide inches per minute by 39.374 (ex. 300 inches per minute divided by 39.374 = 7.62 meters per minute)

RPM Selection - the general operating RPM for tooling contained in this catalog is between 10,000 and 20,000 revolutions per minute. Usually the higher the RPM, the better surface finish becomes. However, the higher the RPM, the higher the friction generated between the tool and the work piece. This friction is what creates the mechanical wear on the cutting edge. Your goal is to select the lowest RPM possible for each application.