FACULTY OF TECHNOLOGY

DELTA ROBOT MOTION CONTROL

Olli Lustig

Supervisor: Toni Liedes

Degree Programme in Mechanical Engineering

Master's thesis 2020

ABSTRACT

Delta robot motion control

Olli Lustig

University of Oulu, Degree Programme of Mechanical Engineering

Master's thesis 2020, 68 p. + 3 p. Appendixes

Supervisor at the university: Toni Liedes

The aim of this thesis is to generate a functional motion control to a delta robot. The motion control is based on solving the inverse kinematics problem of the delta robot. This solution is then used to form the control logic of the robot. In addition, this thesis also introduces forward kinematics solution models and, the most common industrial robots and their features. Applications of industrial robots, as well as the industries that utilize them the most are also examined.

This thesis introduces a self-made delta robot and its motion control design. The functionality of motion control is studied by measuring the positioning accuracy as well as the repeatability of the self-made delta robot in the xy-plane. Accuracy measurements are performed using a separate measuring device. A small-scale comparison between the positioning accuracy of a self-made and a commercial delta robot is implemented to find out how closely can the performance of a commercial delta robot be reproduced with a self-made delta robot.

The results of this thesis indicate that the inverse kinematics model of the delta robot as well as the motion control actually work. The results demonstrate that the performance of the self-made delta robot is at a good level and that further development is worthwhile. There was not enough measurement data to perform a proper comparison between the self-made and the commercial delta robot. However, despite the narrow sampling, it is assumed that the positioning accuracy of the self-made delta robot is not yet at the same level as that of the commercial product.

The accuracy of the self-made delta robot presented in this thesis can be improved by developing the feeding of the robot's drive commands. The materials used in the construction of the robot as well as the quality of the joints also affect the accuracy.

The inverse kinematics model of the delta robot presented in this thesis can be easily scaled to different sized delta robots depending on the application. Motion control can be utilized in the control of delta robots implemented with a similar mechanical structure.

Keywords: Delta robot, motion control

TIIVISTELMÄ

Delta-robotin liikkeenohjaus

Olli Lustig

Oulun yliopisto, Konetekniikan koulutusohjelma

Diplomityö 2020, 68 s. + 3 s. liitteitä

Työn ohjaaja yliopistolla: Toni Liedes

Tämän työn tarkoituksena on suunnitella delta-robotille toimiva liikkeenohjaus. Liikkeenohjauksen rakentaminen perustuu delta-robotin käänteiskinematiikan ratkaisemiseen. Käänteiskinematiikan ratkaisua hyödynnetään ohjauslogiikan toteutuksessa. Työssä tutustutaan myös suorankinematiikan ratkaisumalleihin, sekä esitellään yleisimpiä teollisuusrobotteja ja niiden ominaisuuksia. Työssä tarkastellaan myös teollisuusrobottien käyttökohteita, sekä niitä eniten hyödyntävät teollisuudenalat.

Työssä tutustutaan omavalmisteiseen delta-robottiin ja sen liikkeenohjauksen suunnitteluun. Liikkeenohjauksen toimivuutta tutkitaan mittaamalla omavalmisteisen delta-robotin paikoitustarkkuus, sekä toistotarkkuus xy-tasossa. Tarkkuusmittaukset toteutetaan käyttämällä erillistä mittalaitetta. Työssä pyritään myös selvittämään kuinka lähelle kaupallisen delta-robotin suorituskykyä voidaan päästä omavalmisteisella delta-robotilla. Työssä toteutetaan pienimuotoinen vertailu omavalmisteisen ja kaupallisen delta-robotin paikoitustarkkuuden välillä.

Työn tulokset osoittavat, että delta-robotin käänteiskinematiikan malli, sekä liikkeenohjaus toimivat. Tuloksista selviää, että omavalmisteisen delta-robotin suorituskyky on hyvällä tasolla ja sen kehittämistä kannattaa jatkaa. Omavalmisteisen ja kaupallisen delta-robotin kunnolliseen vertailuun ei saatu riittävästi dataa. Suppeasta otannasta huolimatta on kuitenkin oletettavaa, että omavalmisteisen delta-robotin paikoitustarkkuus ei vielä yllä samalle tasolle kaupallisen tuotteen kanssa.

Työssä esitellyn omavalmisteisen delta-robotin tarkkuutta saadaan parannettua kehittämällä robotin ajokomentojen syöttämistä. Myös robotin rakenteessa käytetyt materiaalit, sekä nivelten laadukkuus vaikuttavat tarkkuuteen.

Työssä esitetty delta-robotin käänteiskinematiikan malli on helposti skaalattavissa myös erikokoisiin delta-robotteihin käyttökohteesta riippuen. Liikkeenohjausta voidaan hyödyntää vastaavalla mekaanisella rakenteella toteutettujen delta-robottien ohjauksessa.

Avainsanat: Delta-robotti, liikkeenohjaus

PREFACE

This master´s thesis was ordered by JOT Automation Oy and the purpose of this work is to develop a functional motion control to a delta robot.

I would like to give warmest thanks to JOT Automation Oy and Juha Saaranen for offering me such an interesting topic to my master's thesis. I would also like to give warmest thanks to Toni Liedes as being a supervisor with a solution to everything. Lastly, I want to thank Mika Muurinen, Henri Remes and Tuomo Kamula for helping in the designing work of the delta robot.

Finally, I want to thank my closest ones for supporting me during my studies.

Oulu, 01.06.2020

Olli Lustig

TABLE OF CONTENTS

APPENDIXES:

Appendix 1. Solving the unknown joint variables from the inverse kinematics equations.

Appendix 2. The MATLAB script for inverse kinematics equations.

SYMBOLS AND ACRONYMS

A_p	Positioning accuracy
a_{i-1}	Transition about $\hat{x}_{i-1}$-axis
c	Cosine
d_i	Transition about axis $\hat{z}$
e	Euler's number
f_i	Number of DOF provided by the i^{th} joint
J	Number of joints
$\bar{l}$	Mean value of positioning repeatability
l_i	i^{th} positioning repeatability
m	Degrees of freedom of a rigid body
N	Number of links
n	Number of measured points
R_P	Positioning repeatability
S	Screw axis
s	Sine
S_l	Standard deviation for l
T_n^0	Homogeneous transformation matrix from n-frame to 0-frame
$\bar{x}$	Mean of measured positions in x-direction
$\hat{x}$	Rotation axis x
x_c	Programmed target position in x-direction
x_i	i^{th} measured x position
$\bar{y}$	Mean of measured positions in y-direction
y_i	i^{th} measured y position
y_c	Programmed target position in y-direction
$\hat{z}$	Rotation axis z
α_i	Angle variable in delta robot kinematics
α_{i-1}	Rotation angle in DH-parameters
γ_i	Angle variable in delta robot kinematics
θ_i	Layout angle of the i^{th} linear screw rail
ϕ	Rotation about the screw axis in degrees
ϕ_i	Rotation about $\hat{z}_i$-axis

AR	Augmented Reality
CNC	Computerized Numerical Control
DH	Denavit-Hartenberg
DOF	Degrees Of Freedom
FSR	Force Sensing Resistor
IP	Internet Protocol
NC	Numerical Control
NCI	Numerical Control Interpolation
OLP	Off-Line Programming
PC	Personal Computer
PLC	Programmable Logic Controller
PoE	Product of Exponentials
SCARA	Selective Compliance Robotic Assembly Arm /Articulated Robot Arm
SME	Small to Medium sized Enterprise
ST	Structured Text
TPM	Translational Parallel Manipulator
VR	Virtual Reality

1 INTRODUCTION

In this thesis the motion control of a delta robot is studied. The base of this thesis is in solving the inverse kinematics model of the delta robot. After the inverse kinematics model is solved it is shown how it can be implemented in the control logic of the delta robot. This thesis demostrate how the control logic of the delta robot can be implemented by using Beckhoff´s programmable logic controller (PLC).

The delta robot is an industrial robot, which was invented in the 80´s by Reymond Clavel and his research team. The invention took place in the Swiss Federal Institute of Technology in Lausanne. The basic idea of the delta robot is to create three translational degrees of freedom using parallelograms. The delta robot is designed for fast pick and place duties. The delta robot has an ability of up to 12 G acceleration in industrial applications. Because of this the delta robot is the best choice for pick and place duties (Bonev, 2001).

The aim of this study is to find the inverse kinematics model of the delta robot and to form a PLC program based on it. Another target of this thesis is to examine the positioning accuracy and repeatability of the delta robot. The positioning accuracy and repeatability are verified by using a grid encoder. The grid encoder is a testing device for machine tool inspection. The delta robot is programmed to make interpolated path movements from point to point. These examinations are carried out with a case study delta robot, which was designed and manufactured earlier by the author of this thesis, Mika Muurinen and Henri Remes during an internship period at JOT Automation Oy.

The positioning accuracy of ABB's Flexpicker IRB 360-1/1130 delta robot is also measured to form a baseline for comparison purposes. It is expected that the commercial delta robot is more accurate and, thus, will yield better results. It is, however, interesting to see how close the results of the self-made delta robot can get to those of the commercial robot. Results comparing the efficiency of a commercial delta robot and a self-made delta robot are presented later in this thesis.

Chapters two to four are based on a literature review. Chapter two includes discussion of the most common industrial robots and their features on a general level. This is followed

by a more theoretical chapter three about the kinematics of serial structured mechanisms. Chapter four focuses on delta robots. Their different mechanical structures as well as the kinematics of the delta robot are discussed. In addition, the solution to the inverse kinematics problem of a delta robot is shown.

Chapters five to seven focus on the case study delta robot. This case study begins by introducing the case study delta robot to the reader of this thesis. Moreover the control logic of the case study delta robot is discussed. After this the performance of the case study delta robot is verified using methods shown in chapter six. Chapter seven introduces the results of the positioning accuracy and repeatability of the case study delta robot. The measurement results are then analyzed and discussed. Lastly a small-scale comparison is made between the case study delta robot and ABB's Flexpicker.

2 INDUSTRIAL ROBOTS

In this chapter the definition of an industrial robot and the most common types are introduced. Moreover, the industrial robots markets are reviewed. After this the degrees of freedom of a robot are discussed before a discussion about accuracy and repeatability. At the end of this chapter the different programming methods for industrial robots are compared.

2.1 Definition

ISO 8373:2012 standard defines the industrial robot as an automatically controlled and reprogrammable device. According to the definition an industrial robot is a multipurpose manipulator that can be programmed in three or more axes. An industrial robot can be either fixed in place or fully mobile and it operates in industrial automation applications. According to ISO 8373:2012 industrial robot includes the manipulator including actuators. Also a controller including a teach pendant is included for programming and controlling purposes (ISO, 2012).

The most popular industrial robot types are articulated robot, Cartesian robot, delta robot and SCARA robot. These robots can be placed under two main topics, which are serial and parallel robots. The two most popular serial structured robots are the articulated robot and the SCARA robot (Selective Compliance Robotic Assembly Arm /Articulated Robot Arm). Articulated robots can be used in various tasks such as arc welding (Doan and Lin 2017), machining (Chen and Dong 2013) or assembly duties (Ranz et al. 2017). Articulated robots use only rotary joints, which makes it possible to seal the joints properly. Because of this ability, articulated robots are perhaps the best choice to work in such environments where it is required that the robot does not pollute the environment (Pan et al. 2017) or even under water (Barbieri et al. 2018). Serial structured robots like articulated robots have often a good working envelope to footprint ratio. Articulated robots can also be mounted to a prismatic axis which enlarges the workspace even more. Articulated robots can also be mounted to the sealing or on the wall, which reduces the footprint even more or completely depending on the case.

One disadvantage of serial structured robots is that their stiffness reduces when more axes are added to the structure, which causes more inertia. Also the tolerance error in the serial robot link manufacturing adds up the total error in the end effector. The SCARA robot is mostly used in quick assembly or material handling. It has a cylindrical work space.

The second main group in industrial robots are the parallel structured robots. The parallel structured robot has at least one closed loop in their kinematic chain. Perhaps the most known parallel robots are the delta robot and the Cartesian robot. The main disadvantage in parallel structured robots is that they suffer from reduced working envelope to footprint ratio. Because of the closed loop chain parallel robots cannot achieve as big a working space as same sized serial structured robots. The biggest advantage in parallel structured robots is that their structure has high stiffness. Also the tolerance errors in parallel robots limbs manufacturing do not add up like in serial robots. The delta robots are mostly used in pick and place tasks and their advantages lie in high speed and acceleration.

The Cartesian robot has a rectangular shaped working envelope and is often used in pick and place, material handling and assembly duties. Cartesian robot has linear actuators for every three Cartesian direction x, y and z. The control logic in Cartesian robots is quite simple because the actuator moves in the same direction with the end effector.

Another popular parallel structured robot is the Stewart Platform, which is often referred to as the Hexapod. Stewart Platform has 6 degrees of freedom so it can rotate and translate about all three axis. This ability makes it possible to use the Stewart platform in flight simulators (Dongsu and Hongbin 2007) , telescopes (Koch et al. 2009) or to compensate the movements of a ship (Zheng et al. 2015)

2.2 Industrial robot markets

The top five markets in industrial robot business are in China, Japan, United States, Republic of Korea and Germany and 74 % of all industrial robots are located in these countries. China was the biggest market for industrial robots in 2018 and accounted for 36 % of all industrial robot installations. This has been an ongoing trend in China since 2013. The peak value in industrial robot installation in China was reached in 2017 and there was a slight drop of 1 % in 2018. However the total amount of 154 thousand units

in China was still more than the number of Europe and USA combined. Industrial robot installations in Japan saw a 21 % growth in 2018 to 55 thousand units. Japan is already a country with a high level automation industry so the average growth rate of 17 % since 2013 is impressive. In the US the total amount of industrial robots has grown for eight years in a row. In 2018 the total amount of industrial robots in the US was over 40 thousand, which was reached by a 22 % increase compared to the previous year. Also in the US the growing need for robots is due to added automation in manufacturing processes. The need for industrial robots in the Republic of Korea is generated by the electronics industry. Since the peak level of 41 thousand industrial robots in 2016 the number of units has decreased. In 2018 the total amount of industrial robots was just under 38 thousand, which was 5 % less than in 2016. Germany reaches the fifth place in the total amount of industrial robots in the world, with its 26 thousand units. There was an increase of 26 % compared to the previous year in 2018. The need for such a high level of industrial robots in Germany follows from the automotive industry (International federation of robotics, Executive Summary WR 2019 Industrial Robots, 2020). The total amount of industrial robot installations by regions can be seen in figure 1.

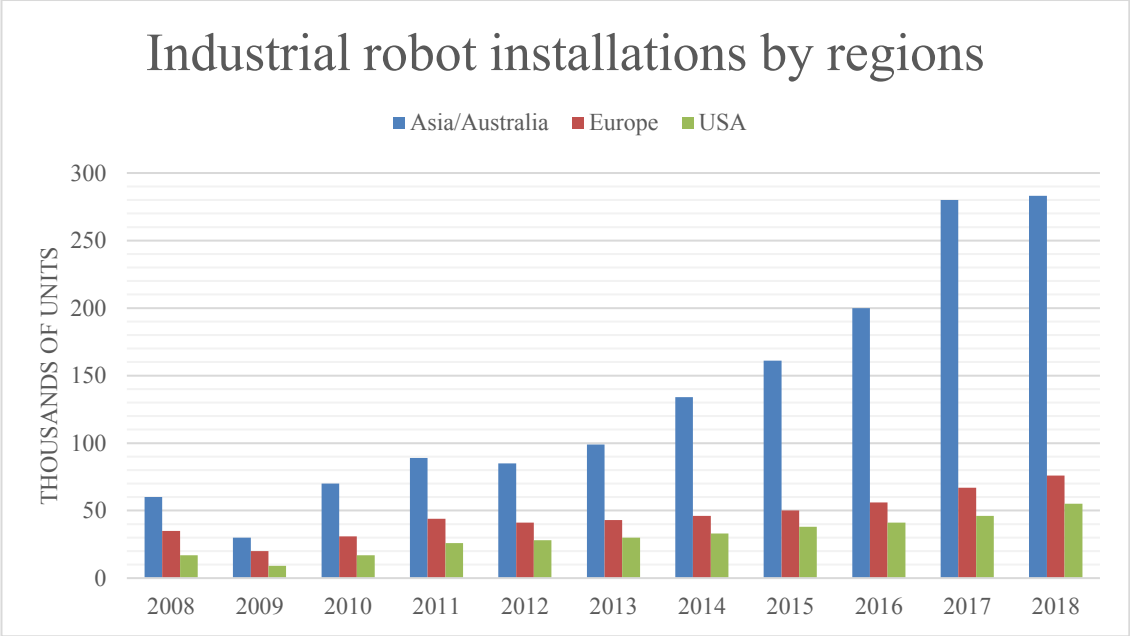

Figure 1. The total amount of industrial robot installations by regions.

Most of the industrial robots are in some way connected to the automotive industry. Almost 30 % of industrial robots work in this area of business. The number of industrial robots has seen an annual growth rate of 13 % in 2013–2018. The increased level in robots is because of the economic crisis in 2009. After 2009 the automotive industry has changed its ways of working and increased the automation level in the manufacturing processes.

The second largest market for industrial robots is in the electric and electronic industry. Manufacturing of communication devices, computers, radios and TV´s are included in this category. Also precision and optical products as well as medical equipment are included in the electronic industry. In 2017 31 % of industrial robots were linked to this area of business and the number of installed units almost passed the corresponding number of the automotive industry. In 2018 the need for electronic components decreased and the amount of installed industrial robots dropped by 14 %. The need for electronic components was affected by the trade crisis between USA and China.

2.3 Degrees of freedom

Robot´s configuration describes in which pose the robot is at the moment. Since a robot consist of links of a known shape, only a few parameters are needed to specify the robot´s configuration. For every robot the least amount of these real numbers that describe the configuration can be determined. The number of degrees of freedom tells this minimum amount of parameters, that are needed to represent the configuration (Lynch and Park 2017, p. 29). The degrees of freedom (DOF) of a mechanism can be discussed through a simple door example. If a door is not attached to a door frame by a hinge joint, the door is able to rotate and translate about all three axes and would thereby have six degrees of freedom in total. Using the hinge joint reduces the degrees of freedom of the door and it can only rotate about the joint axis (Lynch and Park 2017, p. 33). By this example it is obvious that the joints used in the mechanism has an effect on the degrees of freedom. There are two other joint types that have only one DOF. The prismatic joint allows only translational movement about the joint axis and the helical joint, which allows translation and rotation about the screw axis. There are also joints that have multiple DOF. The cylindrical joint allows rotational and translational movement about the joint axis. The universal joint has two rotational degrees of freedom and a spherical joint allows movement in all three rotational axes (Lynch and Park 2017, p. 34).

These joints are used in every moving mechanism to connect the links. Degrees of freedom of a mechanism depends on the number of joints and links that are being used and can be calculated using Grübler's formula, which is shown in equation (1), (2) and (3) (Lynch and Park 2017, p. 35).

$$\mathrm{DOF} = m(N - 1 - J) + \sum_{i=1}^{J} f_i \quad (1)$$

,where m is the degrees of freedom of a rigid body,
N is the number of links,
J is the number of joints,
f_i is the number of DOF provided by the i^{th} joint.

Using this formula to the door example above gives:

$$\mathrm{DOF} = 3(2 - 1 - 1) + 1 = 1 \quad (2)$$

,which means that the door would have one degree of freedom just like discussed before. The number of degrees of freedom can be either a positive or negative integer or zero (Duysinx and Géradin 2004, p. 28). If the DOF of a mechanism is greater than zero the mechanism is considered as a mobile mechanism. When the DOF of the mechanism is equal to zero the mechanism is a statically determined structure. This means that the mechanism is unable to move at all. The DOF can also be a negative integer and in these cases the mechanism is considered as a statically undetermined mechanism or an over constrained mechanism (Stamper 1997, p. 31). If we take another look at the door example discussed by Lynch and Park (2017, p. 33) and add another hinge joint to the mechanism, we can see that the results from Grübler's formula are not always that easy to utilize. Adding another hinge to the door mechanism gives:

$$\mathrm{DOF} = 3(2 - 1 - 2) + 2 = -1 \quad (3)$$

As mentioned before when the DOF gets a negative value the mechanism is a statically undetermined structure. However, it is clear that the door can actually rotate about the joint axis no matter how many hinges there are. As Lynch and Park mentioned (2017, p.

39) Grübler's formula requires that the constraints provided by the joints have to be independent which is not always achieved.

Another downside of the Grübler's formula is that it only tells the total number of the degrees of freedom. It would be useful to know whether these degrees of freedom are translational or rotational (Li, Xu 2007a, p. 3).

2.4 Accuracy and repeatability

Robot's performance is measured in three parameters. Accuracy describes how precisely the robot can move it´s end effector to programmed position and orientation. The second parameter is repeatability, which is perhaps even more important when comparing different robots. Repeatability describes the robot's ability to always reach the same position and orientation. For example, if the robot's end effector positions vary 5 mm from the programmed position, the robot's accuracy is quite poor. However, if the robot always reaches the same 5 mm inaccurate position, its repeatability is excellent. The third performance parameter is resolution. Resolution is used to describe the smallest increment in the end effector position and orientation the robot can produce (Conrad and Shiakolas 2000). Resolution is limited by the resolution of position and velocity sensors in the joints. Also the gear ratio of the joints affects the resolution (Duysinx and Géradin 2004, p. 48).

There are multiple things that affect the overall accuracy of a robot. According to Conrad and Shiakolas (2000) these accuracy degreasing aspects can be divided into three main groups. Firstly, the dynamics of the robot affect its overall accuracy. Because robots can move quite fast, inertia has a great effect on performance. There is friction in the joints and also the servo system including encoders have limited resolution. The second main group consists of structure related concerns. Operating temperature and payload causes many drawbacks in accuracy. Also the bearings and gears cause friction. The third group is the kinematics. The forward kinematics problem is solved with DH (Denavit Hartenberg) -parameters of the robot. These parameters include link lengths and coordinate transformations that are calculated from the robot´s mechanical geometry. All the tolerance error in link manufacturing causes inaccuracy. The difference between the

mechanical dimensions and the forward kinematics formula was also discussed by Siciliano et al. (2009, p. 108).

Also the mechanical structure of the robot affects its accuracy. According to Siciliano et al. (2009, p. 26) a Cartesian robot has a constant accuracy in every point of the work space. This is because every degree of freedom of the robot correspond to a Cartesian variable. This means that every parallel actuator moves the end effector only in x-, y- or z-direction. In comparison to a cylindrical or a spherical manipulator the accuracy differs inside the work space. The usage of one or multiple rotational joints causes this error in accuracy. Because the resolution in a rotary encoder is constant, the accuracy decreases as the radial stroke increases.

The positioning accuracy A_p is calculated using equation (4) based on the ISO 9283 standard, which was also used by Şirinterlikçi et al (2009). Also Płaczek and Piszczek (2018) and Stephan et al. (2009) used the newer version of the same standard when evaluating the accuracy and repeatability of an industrial robot. Figure 2 presents the positioning accuracy and repeatability. As equation (4) and figure 2 show, positioning accuracy can be calculated from the mean value of attained positions and the programmed target position. The positioning accuracy is the minimum distance between the mean value of all measured points and the target point.

$$A_p = \sqrt{(\bar{x} - x_c)^2 + (\bar{y} - y_c)^2} \tag{4}$$

,where A_p is the positioning accuracy,
$\bar{x}$ is the mean of measured positions in x-direction,
x_c is the programmed target position in x-direction,
$\bar{y}$ is the mean of measured positions in y-direction,
y_c is the programmed target position in y-direction.

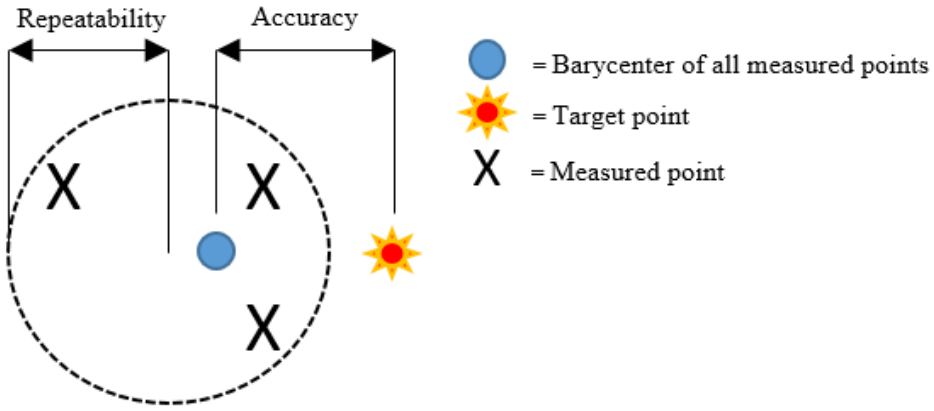

Figure 2. Positioning accuracy and repeatability.

The positioning repeatability is the radius of a smallest enclosing circle of all measured points, which can be seen from figure 2. The positioning repeatability can be calculated using the following equations (5) (6) and (7). At first the mean value of positioning repeatability is calculated using equations (5) and (6). After this the standard deviation can be defined using equation (7).

$$l_i = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} \tag{5}$$

$$\bar{l} = \frac{1}{n}\sum_{i=1}^{n} l_i \tag{6}$$

$$S_l = \sqrt{\frac{\sum_{i=1}^{n}(l_i - \bar{l})^2}{n-1}} \tag{7}$$

,where l_i is the i^{th} positioning repeatability,
x_i is the i^{th} measured x position,
x_c is the target x position,
y_i is the i^{th} measured y position,
y_c is the target y position,
$\bar{l}$ is the mean value of positioning repeatability,
n is the number of measured points,
S_l is the standard deviation for l

The positioning repeatability can be calculated using equation (8) according to ISO 9283:1998 standard.

$$R_P = \bar{l} + 3 \cdot S_l \tag{8}$$

,where R_P is the positioning repeatability,
$\bar{l}$ is the mean oppositional repeatability,
S_l is the standard deviation for l.

2.5 Programming methods

Using robots in production processes increases productivity and flexibility of a company. However, productivity can increase only when lot sizes are big enough to cover the costs caused by robot programming. Consequently, small to medium sized enterprises (SME) cannot easily profit from investing in a robot. According to Pan et al. (2012) in a vehicle hull welding process it can take up to 8 months to program the correct trajectories for the robot. The welding process itself takes only 16 hours so the time consumed in teaching the robot is about 360 times higher.

There are two main methods for industrial robot programming. In online programming, which covers lead-through- and walk through –programming, the robot's end effector is manually moved to different positions which are then saved in the robot's memory. Moving the robot is done with a teach pendant by the operator. These positions can later be used in programming the robot. The basic principle of online programming method is simple but it can be utilized only in simple processes and the quality of the product is highly dependent on the operator´s skills (Pan et al. 2012). The Lead-through method or jogging the robot from position to another is not usually that intuitive and it takes a lot of effort from the programmer to manage between multiple coordinate systems. When the programming is finally completed there is still a lot of testing and verifying before the robot can be used in production safely. In addition to time consuming programming, the reuse of the program is difficult. Even if there are only a few differences in the workpiece, the programming work has to start over. Despite all of these downsides, online programming is usually the only option the SME companies have (Pan et al. 2012).

There have been multiple methods to improve online programming. Sugita et al (2004) presented an option to traditional teaching pendant for programming a deburring robot. This three-wire type teaching support device includes a dummy tool that can be moved by hand. The position and orientation of the tool can be measured and this data can be used later in programming the actual robot. Sugita et al (2004) also presented an arm-type teaching support device, which achieved more accurate measurements in comparison to the wire type teaching support device.

Schraft and Meyer (2006) introduced a new online programming method especially for SMEs with small lot sizes. This method includes a force and torque sensing handle that is mounted directly on the robot. In this walk-through method the force applied to the handle by the user is measured and this way the movements of the robot can be generated. This type of walk-through method has been used for a long time in industry but the trajectories generated by human hand have not been accurate enough for some processes and too hard to modify. To overcome this problem Schraft and Meyer (2006) introduced tools to modify the trajectories and velocity afterwards.

Also Choi and Lee (2001) introduced a walk-through programming method, which was based on a force and torque sensing handle. This COSMO-II sensor can be used to program a 6-DOF robot arm. The principle of the COSMO-II sensor is based on Force Sensing Resistors (FSR) that detect the movements of the handle. When force is applied to the handle the contact bolts inside it are being pressed closer to FSRs and the resistance of it changes.

There are also multiple online programming methods that are not affected by the inaccuracy of a human. Zhang et al. (2006) introduced an automated online programming method for a 6 DOF robot used in deburring aluminum wheels. This programming method benefits from different sensors and is therefore not affected by the inaccuracy of a programmer. This programming method uses a camera and visual processing of the image to follow a marked line on the wheel. The camera is used to define the XY coordinate of the tool tip. In addition to the visual sensing a force sensor is used. The tool tip is kept in contact with the surface utilizing the measurement data from the force sensor. This way the missing Z coordinate is generated. According to Zhang et.al (2006) this sort

of sensor guided online programming method for wheel deburring reduces the programming time to less than an hour.

Offline programming (OLP) method is feasible when production lot sizes are large enough. In OLP method the whole robot sell is modelled in 3D CAD world. In the 3D simulation program, it is easy for the programmer to test all trajectories of the robot and to ensure that they are collision free. The biggest advantage of using OLP method is that the robot can work while the next program is generated, which decreases the robot down time. When planning a new production cycle, robot programming can be done at the same time with the rest of the production work and not after it like in online programming. It is also easy to modify the trajectories and points in the robot program, which increases the reusability of an old robot program. As mentioned before the OLP method is financially reasonable only with lot sizes that are large enough. Hence, SMEs are usually not able to benefit from offline programming. OLP software is expensive and usually some modifications are required before it is able to reach the requirements of a process. These modifications to the software require high level programming that usually exceeds the skill level of a traditional robot programmer. All of these costs can be seen in increased programming overhead. Most of the time the 3D model of the robot sell does not match the real world and some modifications have to be done to the robot program. To meet the accuracy requirements of the process some calibration to the trajectories have to be done using different sensors.

Offline programming begins by generating the 3D model of the robot cell and all the components in it. Usually there already exists a 3D model for most of the components but a 3D model can also be generated using a laser scanner like Bi and Lang presented (Bi, Lang 2007, p. 4). After the robot sell is modelled in 3D world some position tags are generated. These tags include features from the workpiece like corners and edges and robot tool position that help in the programming phase, like robot home position, or approach points. A serial robot can reach a point in space with multiple configurations. It is the programmer's job to decide which configurations are used in program and how to minimize the transition from configuration to another. If there are multiple robots working on the same workpiece the process timing needs to be discussed to minimize the cycle time. After this the robot cell actions can be simulated and robot trajectories can be confirmed without using the real robots. Now the program can be transferred to the actual

robot and the calibration work begins. Ideally the program would work immediately in the real world but in most cases it has to be fine-tuned.

Yet another option to robot programming is based on augmented reality (AR). Robot programming using AR is the newest programming method and it is based on virtual robot in the real world. AR utilizes computer generated objects blended with the real world. AR is used to generate robot trajectories and it is very intuitive just like the walk-through method. Using AR, the programmer can choose between multiple simulated trajectories and they can be modified afterwards. In addition, AR programming is much safer than online programming. It is intuitive because the programmer can move the robot in augmented reality. AR makes robot's trajectories and the surrounding area scalable, which helps when programming bigger robots like a robot used to wash an airplane. The biggest advantage in the AR in comparison to VR (virtual reality) is that in AR the surrounding area does not have to be simulated (Chong et al. 2009). OLP programming method using VR was discussed also by Holubek et al. (2018).

3 KINEMATICS

Industrial robots can be divided into two groups depending on their mechanical structure. A serial structured robot is a robot which links are attached to the next and previous with a motor-actuated joint. A serial structured mechanism is also known as an open-chain mechanism (Lynch and Park 2017, p. 36). In an open-chain mechanism the first link is attached to the stationary base and the last link holds the end effector. An articulated robot is an example of a serial structured robot.

The other type of industrial robots are the parallel structured robots. The parallel robots consist of at least one closed loop in their mechanical structure (Lynch and Park 2017, p. 263). A good example of a parallel robot is the Stewart Platform where six actuators are attached to the same platform. A delta robot is also a good example of a parallel mechanism.

Generally said, for serial structured robots the forward kinematics problem is much easier to solve in comparison to parallel structures. When solving the forward kinematics problem of a parallel structure there might be multiple or no solutions. On the other hand, the inverse kinematics problem is much more complex for serial structures (Lynch and Park 2017, p. 247). With parallel structures the inverse kinematics solution is dependent on the number of closed kinematic loops in the mechanism. For parallel structures the inverse kinematics problem gets easier when the number of closed loops increases (Stamper 1997, p. 33). For example, the most common version of the Stewart platform includes six closed loops in its structure, while the corresponding number for the delta robot is only three.

In this chapter the forward and inverse kinematics of a serial structure are discussed. The forward kinematics problem is reviewed using two popular methods, the Denavit-Hartenberg method and the product of exponentials.

3.1 Forward kinematics in an open chain mechanism

Forward kinematics in robotics are used to define the location and orientation of the end effector from varying joint positions (Lynch and Park 2017, p. 137). The forward

kinematic problem for an open chain mechanism can be solved with two different methods. The first one is called the Denavit-Hartenberg method where reference frames are used to solve the forward kinematics problem. In the Denavit-Hartenberg method the transition from reference frame to another can be done by using four parameters, which are called the DH-parameters of the mechanism. Three of these parameters describe the transition from reference frame to another and the fourth parameter defines the varying joint position θ. Four is the minimum amount of parameters to describe the displacement between two link frames (Lynch and Park 2017, p. 176).

Defining the displacement between link frames i-1 and i can be done as shown in figure 3. The first thing to do is to set the $\hat{z}_{i\text{-}1}$-axis of the reference frame i-1 coincident with the rotation axis i-1. This can be done by rotating the i-1 frame about the $\hat{x}_{i\text{-}1}$-axis by $\alpha_{i\text{-}1}$ degrees. After this the shortest distance $a_{i\text{-}1}$ between axis $\hat{z}_{i\text{-}1}$ and $\hat{z}_i$ is defined. Now the reference frame i-1 can move along $\hat{x}_{i\text{-}1}$-axis for $a_{i\text{-}1}$ units. After this the reference frame i-1 is moved for d_i units about the $\hat{z}_i$ -axis. The last thing to do is to rotate the i-1 frame about the $\hat{z}_{i\text{-}1}$-axis by ϕ_i degrees to make reference frames i-1 and i corresponding (Lynch and Park 2017, p. 604).

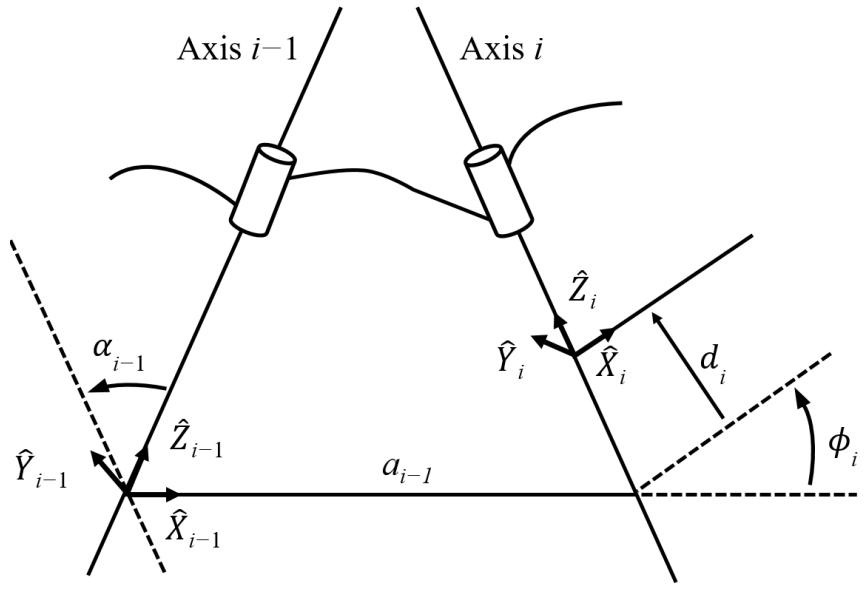

Figure 3. Moving from a reference frame to another (retell Lynch and Park 2017).

After transformations for every frame is defined according to DH-parameters the forward kinematics solution is solved by multiplying them sequentially. For every homogeneous transformation $T_{i-1,i}$ can be written as shown in the equations (9)−(14) (Lynch and Park 2017, p. 608).

$$T_{i-1,i} = \mathrm{Rot}(\hat{x}, \alpha_{i-1})\ \mathrm{Trans}(\hat{x}, a_{i-1})\ \mathrm{Trans}(\hat{z}, d_i)\ \mathrm{Rot}(\hat{z}, \phi_i) \tag{9}$$

$$T_{i-1,i} = \begin{bmatrix} \cos\phi_i & -\sin\phi_i & 0 & a_{i-1} \\ \sin\phi_i \cos\alpha_{i-1} & \cos\phi_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -di \sin\alpha_{i-1} \\ \sin\phi_i \sin\alpha_{i-1} & \cos\phi_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & di \cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{10}$$

where

$$\mathrm{Rot}(\hat{x}, \alpha_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_{i-1} & -\sin\alpha_{i-1} & 0 \\ 0 & \sin\alpha_{i-1} & \cos\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{11}$$

$$\mathrm{Trans}(\hat{x}, a_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{12}$$

$$\mathrm{Trans}(\hat{z}, d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{13}$$

$$\mathrm{Rot}(\hat{z}, \phi_i) = \begin{bmatrix} \cos\phi_i - 1 & -\sin\phi_i - 1 & 0 & 0 \\ \sin\phi_i - 1 & \cos\phi_i - 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

,where a_{i-1} is the transition along $\hat{x}_{i-1}$-axis,

$\hat{x}$ is the rotation axis x,

$\hat{z}$ is the rotation axis z,

α_{i-1} is the rotation angle about rotation axis $\hat{x}_{i-1}$,

ϕ_i is the rotation about $\hat{z}_i$-axis,

d_i is the transitions along rotation axis $\hat{z}$.

Figure 4 shows an example of DH-parameter usage for an open chain mechanism with three revolute joints. The table shown in figure 4 includes the DH parameters for the system. Reference frames 0 and 1 are in the same position and orientation. Therefore, the first line of the table includes only rotation component ϕ_1. As seen from figure 4 the reference frames 1 and 2 are in the same position but have different orientation. To move from reference frame 1 to 2 the reference frame has to be turned about the $\hat{x}_1$-axis. There is also the varying joint angle ϕ_2 on the same row. Reference frames 3 and 4 are in the same orientation as the reference frame 2. Transition from the reference frame 2 to 3 is achieved by moving along $\hat{x}_2$-axis for $L1$ units. Transition from reference frame 3 to 4 is done in the corresponding method for $L2$ units.

i	α_{i-1}	a_{i-1}	d_{i-1}	ϕ_i
1	0	0	0	ϕ_1
2	90°	0	0	ϕ_2
3	0	L1	0	ϕ_3
4	0	L2	0	0

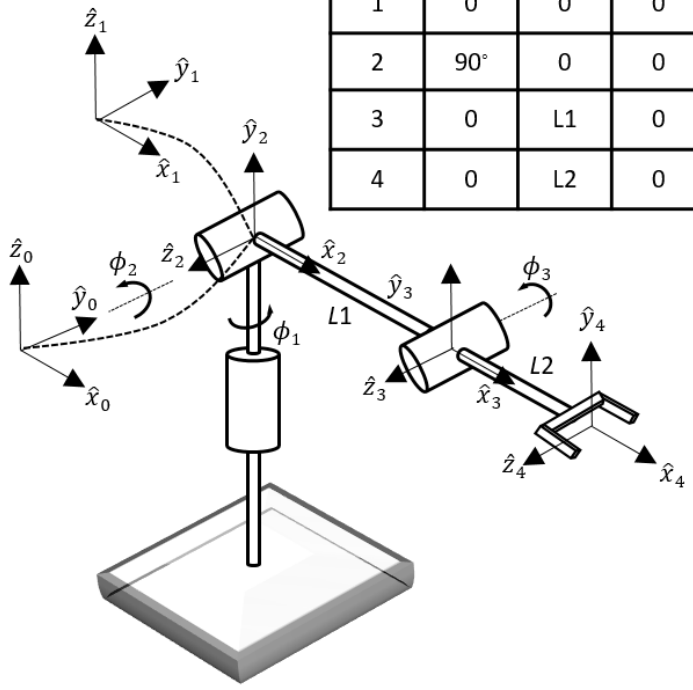

Figure 4. An open chain mechanism with three revolute joints.

Another method for solving the forward kinematics problem of an open chain mechanism is called the product of exponentials (PoE). The PoE method was reviewed by Lynch and Park (2017, p. 140-142). The PoE method begins by defining a fixed frame {s} in the robot´s base. Another frame {b} is located at the tip of the last link of the robot. After this

the robot is considered in its “home” or “zero” position. The home position of the robot is when it is in a configuration where all joint values are known. The programmer decides which configuration is to be used as a home position.

The second phase of the PoE method is to define the matrix *M* in which the position and orientation of frame {b} in the {s} frame is included when the robot is in home position. The matrix *M* is a 4x4 matrix and tells the x, y and z position and orientation of the frame {b}.

When the robot is in the home position, the screw axis for every revolute joint can be written. The screw axis for all joints can be written as a 1x6 matrix. The first three rows describe which axis of the fixed {s} frame the joint rotates about. For example in the case of a planar xy-mechanism the joint axis is always in the z-axis direction. The last three rows of the matrix describe the distance of the screw axis from the fixed {s} frame. The screw axis is defined for every joint in the same way.

After the screw axis is defined for all joints the screw motion for the whole mechanism can be written as shown in equation (15) (Lynch and Park 2017, p. 200).

$$T(\phi) = e^{[S1]\phi 1} \cdot \ldots \cdot e^{[Sn]\phi n} \cdot M \tag{15}$$

, where $T(\phi)$ is the new configuration of the end-effector frame,
e is the Euler’s number,
S is the screw axis,
ϕ is the rotation about the screw axis in degrees,
M is the 4x4 matrix.

The PoE method can also be applied to the open chain mechanism presented in figure 4. The mechanism is at its home position and the 4x4 matrix *M* can be written as shown in the equation (16).

$$M = \begin{bmatrix} 1 & 0 & 0 & L1 + L2 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

,where M is the 4x4 matrix,
$L1$ is the distance between reference frames 2 and 3,
$L2$ is the distance between reference frames 3 and 4.

As seen from equation (16) the reference frame 4 is moved for $L1 + L2$ units along the $\hat{x}_0$-axis. The reference frame 4 is also rotated about x-axis for 90°.

The screw axes $S_i = (\omega_i, v_i)$ are listed in table (1). The ω_i parameter includes the rotation part of the S_i matrix. For an example $\omega_2 = (0, -1, 0)$ tells that the screw axis of the reference frame 2 rotates in the $-\hat{y}_0$-direction of the reference frame 0. The v_i component in the screw axes can be attained by writing $v_i = -\omega_i \times q_i$, where q_i is any point on joint axis i as written in coordinates in the fixed base frame. (Lynch and Park 2017, p. 142). For example by choosing $q_4 = (L1+L2, 0, 0)$ the v_4 component can be solved by calculating $v_4 = -\omega_4 \times q_4$, which gives $(0, 0, -(L1+L2))$.

Table 1. Screw axes

i	ω_i	v_i
1	(0, 0, 1)	(0, 0, 0)
2	(0, -1, 0)	(0, 0, 0)
3	(0, -1, 0)	(0, 0, -$L1$)
4	(0, -1, 0)	(0, 0, -($L1$+$L2$))

The PoE method can be studied through an example case. We assume that the mechanism shown in figure 4 is in a configuration, where the joints 2 and 3 have both turned by 45° and the gripper is pointing straight upwards. We also assume that the links $L1$ and $L2$ have lengths $L1$ = 550 mm and $L2$ = 450 mm. By doing this the equations (17)–(19) can be written.

$$e^{[S_2]\cdot\pi/4} = \begin{bmatrix} 0{,}71 & 0 & -0{,}71 & 0 \\ 0 & 1 & 0 & 0 \\ 0{,}71 & 0 & 0{,}71 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (17)$$

$$e^{[S_3]\cdot\pi/4} = \begin{bmatrix} 0{,}71 & 0 & -0{,}71 & 161{,}09 \\ 0 & 1 & 0 & 0 \\ 0{,}71 & 0 & 0{,}71 & -388{,}91 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

$$T(\phi) = e^{[S2]\phi 2} \cdot e^{[S3]\phi 3} \cdot M = \begin{bmatrix} 0 & -1 & 0 & 388{,}91 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 838{,}91 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

As can be seen from equation (19) the end-effector has now translated to position x = 388,91 mm, z = 838,91 mm in the fixed base frame. It has also rotated 90° about the $\hat{x}_0$- and $\hat{z}_0$-axes.

3.2 Inverse kinematics

The purpose of defining the inverse kinematics solution to an open chain mechanism is to find joint positions that produce the desired position and orientation of the end effector with respect to the base frame. Siciliano and Khatib (2008, p. 84) introduced a solution method for the inverse kinematics problem of a 6-DOF serial manipulator. Finding the solution to a inverse kinematics problem requires usage of six nonlinear equations. Three of these equations are used to define the inverse position kinematics and the remaining three equations define the orientation of the end effector. Together these equations form the homogeneous transformation matrix T_6^0 which combines the position and orientation information from the nonlinear equations. Notation T_6^0 describes the transformation from the sixth reference frame to the base frame. It is possible that no solution for inverse kinematics is found. There can also be multiple configurations for the joints that produce the desired position and orientation of the end effector. For a solution to be possible the target position has to be located in the working envelope of the mechanism. To find the solution to the inverse kinematics problem of an open chain 6-DOF mechanism, there are also two requirements for the mechanical structure. Three consecutive joints have to intersect in the same point. These intersecting joints are the joints that form the spherical wrist. The second condition is that three consecutive joints have to be parallel to each other. This condition is fulfilled by the first three joint axes. All six joint axes are revolute joints (Siciliano and Khatib 2008, p. 85).

According to Siciliano and Khatib (2008, p. 84) the forward kinematics problem of a 6-DOF mechanism can have either a closed-form solution or then it can be solved by using numerical methods. The closed-form solutions are usually preferred because they are fast to form and they include all possible solutions. The biggest disadvantage in the closed-form solutions is that they are robot dependent and cannot be utilized in general form. The first closed-form solution that Siciliano and Khatib (2008, p. 27) introduce is the algebraic method. In the algebraic method the most significant equations that contain the joint variables are identified and then these equations are manipulated to a more soluble form with only one unknown parameter. The second closed-form solution is the geometric method. In the geometrical method the inverse kinematics problem is divided into two parts. At first an inverse position kinematics is formed and after this the inverse orientation kinematics. The first three revolute joint axes define the position of the end effector and the spherical wrist defines the orientation. Siciliano and Khatib (2008, p. 85) show that there are four possible solutions for the inverse position kinematics. When the two possible solutions for the inverse orientation kinematics are added, there are in total eight solutions for the inverse kinematics problem for a 6-DOF serial robot.

Siciliano and Khatib (2008, p. 28) introduced also numerical methods for solving the inverse kinematics problem. The most valuable thing with numerical solving methods is that they are not robot dependent and so these solution methods can be applied to every mechanical structure. By using numerical methods the rotational and translational equations can be turned into a single equation with only one variable of 16^{th} degree. This means that by using a 16^{th} degree equation for the inverse kinematics problem, there can be sixteen solutions in total. However, a closed-form solution can only exist if the degree of the equation is four or less so in many cases, the mechanism is not soluble in closed-form. Numerical methods can be divided into three categories that are symbolic elimination methods, continuation methods, and iterative methods.

4 DELTA ROBOTS

In this chapter the different mechanical designs of the delta robot are introduced. Also the forward and inverse kinematics of the delta robot are discussed.

4.1 Mechanical designs

Parallel manipulators like the delta robot differ from the more popular serial manipulators due their mechanical structure. Parallel mechanisms consist of at least two kinematic loops that usually connect the moving platform to a stationary base. These kinematic loops are often referred to as “legs” or “limbs” of the robot. There is one actuated joint in every leg and the rest of the joints are passive. This structure ensures that the actuators are stationary and therefore their mass does not have to be moved. In serial robots the usage of actuator in every single joint increases inertia and reduces the payload capacity. Because of this, the major difference between parallel and serial robots is that the limbs of the parallel manipulator can be made much lighter, which guarantees higher accelerations and velocities. The biggest disadvantage in parallel manipulators is the reduced work space in comparison to a same sized serial robots (Stamper 1997, p. 1).

There are two popular designs for delta robots. The difference between these two designs is in the actuator. Perhaps the most popular and known version of the delta uses rotary actuators, like (Stamper 1997) and Baqai et al. (Ghazi et al. 2018) discussed. In these versions the kinematic chain is 3-RSS, which means that there are 3 kinematic chains that consist of one revolute and two spherical joints. The other type of delta robot uses a prismatic actuator. This is usually accomplished by using a linear screw rail that is actuated by a servo motor. Linear delta robots can have different kinematic chains. Li and Xu have presented three different types of delta robots with different kinematic chains including a 3-PRS chain (Li and Xu 2007b), 3-PUU chain (Li and Xu 2008) and a 3-PRC chain mechanism (Li and Xu 2009). No matter which kinematic chain is being used, the parallelogram leg design ensures that the end effector has only 3 translational degrees of freedom and the delta robot can be seen as an x-y-z Cartesian positioning device (Lynch and Park 2017, p. 40).

4.2 Delta robot kinematics

In this chapter the forward and inverse kinematics are discussed. The inverse kinematics model of a delta robot is solved using a vector loop method.

4.2.1 Forward kinematics

The forward kinematics problem is to find the unknown position of the end effector from the known joint values. The forward kinematics problem was discussed by (Stamper 1997). In Stamper´s study the position vector $\bar{p}$ of the end effector position P is solved from the known revolute joint values θ_1, θ_2 and θ_3 in the base xyz-coordinate frame. This mechanism consists of three identical limbs with revolute joint actuators. Each limb is built up by an input link and an upper arm with a four-bar parallelogram structure. In this study the forward kinematics problem is solved by using the loop closure equations for every limb. These equations are reduced algebraically to two 16^{th} degree polynomial equations with two unknown angle variables. After this the first unknown variable is reduced by using the dialytic elimination method, which yields only one 32th degree equation. It is shown in the study that from the 32 possible solutions 16 are extraneous leaving 16 possible solutions for the forward kinematics problem.

Güner et al. 2019 introduced a solution to a forward kinematics problem of a 3-PUU translational parallel manipulator (TPM). In this study the manipulator consists of three identical kinematic loops that have an actuated prismatic joint following by two passive universal joints. The solution of forward kinematics problem begins by finding the DH-parameters for each limb. After this, two vector loops are derived from the fixed base coordinate system to the moving coordinate system. Reference frames for each link are formed and by multiplying them the homogeneous transformation matrix is derived. Using these equations the forward kinematics of a 3-PUU TPM can be solved.

4.2.2 Inverse kinematics

The solution to the delta robot inverse kinematics problem was discussed by Uyar and Mutlu (2012). In their analytic vector-loop method, presented in figure 5, a fixed world coordinate system O($\vec{x}$, $\vec{y}$, $\vec{z}$) is assigned in the center of the upper frame. This upper frame can be seen as a triangle of points A1, A2 and A3. The moving platform is

determined by points B1, B2 and B3 and in the center of this platform is the moving coordinate system P($\vec{u}, \vec{v}, \vec{w}$). All three linear screw rails $\overline{A_\iota C_\iota}$ are in lay out angle of θ degrees. The moving platform is attached to the prismatic joints $\overline{A_\iota C_\iota}$ (linear screw rails) by three limbs $\overline{C_\iota B_\iota}$, where i = 1,2,3.

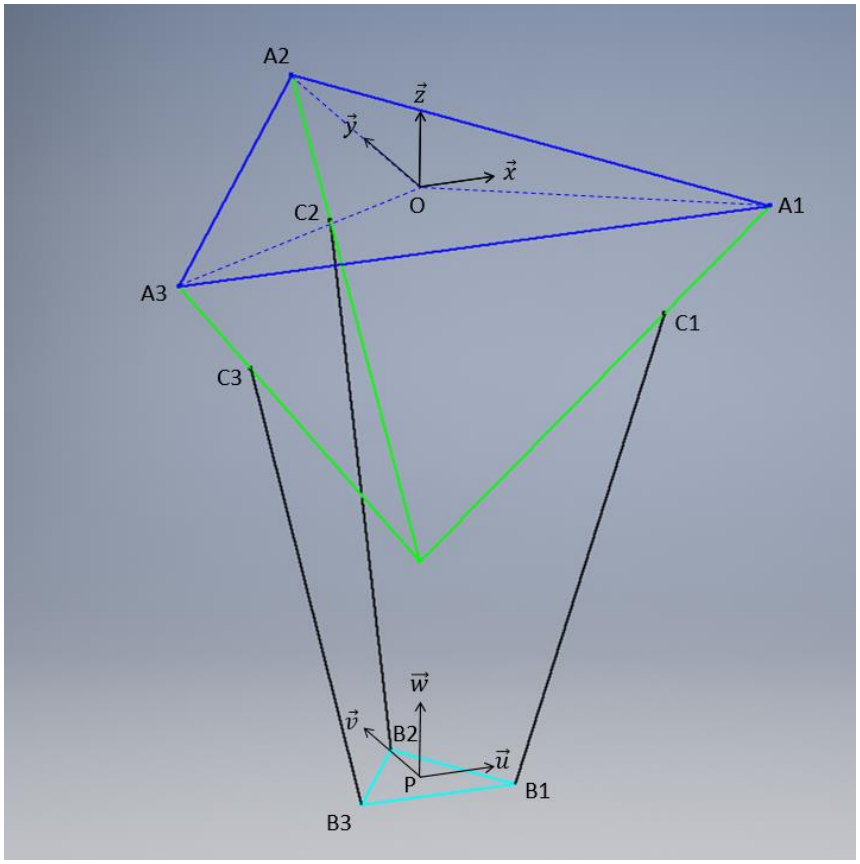

Figure 4. Wireframe model of the delta robot.

In figure 6 the two varying angles α_i and γ_i for each limb are presented. Angle γ_i describes the angle between the limb and the vertical z-axis. Angle α_i is between the projection of vector $\overline{C_\iota B_\iota}$ to the fixed frame and the vector $\overline{OA_\iota}$.

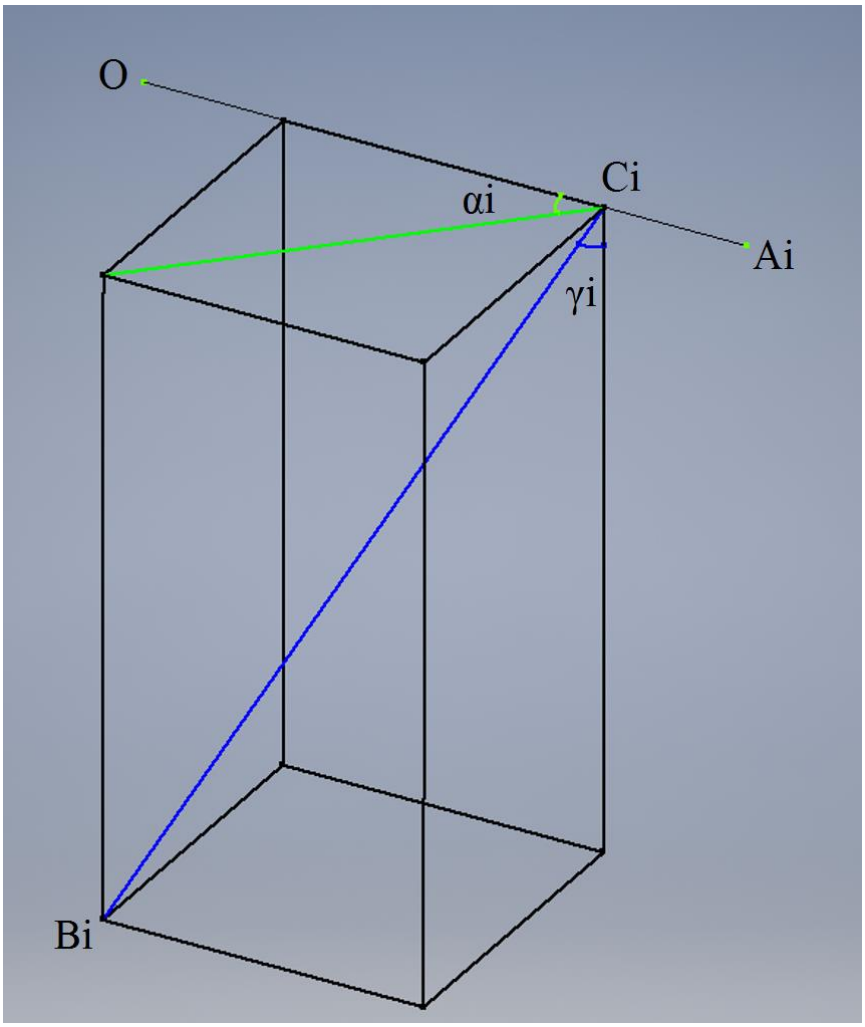

Figure 5. α_i and γ_i angles of the rod.

According to Uyar and Mutlu (2012) vector loops can be used to solve nonlinear kinematic equations (21)–(29). Vector OP can be written using position vectors A_i, B_i and C_i as shown in the equation (20).

$$\overline{OP} = \overline{OA_i} + \overline{A_iC_i} + \overline{C_iB_i} + \overline{B_iP} = +xi + yj + zk \tag{20}$$

$$i_1 \rightarrow \overline{OA_1} \cdot \mathrm{c}(30) - \overline{A_1C_1} \cdot \mathrm{c}(\theta) \cdot \mathrm{c}(30) - \overline{C_1B_1} \cdot \mathrm{s}(\gamma_1) \cdot \mathrm{c}(30 + \alpha_1) - \overline{B_1P} \cdot \mathrm{c}(30) - x = 0 \tag{21}$$

$$j_1 \rightarrow -\overline{OA_1} \cdot s(30) + \overline{A_1C_1} \cdot c(\theta) \cdot s(30) + \overline{C_1B_1} \cdot s(\gamma_1) \cdot s(30 + \alpha_1) + \overline{B_1P} \cdot s(30) - y = 0 \tag{22}$$

$$k_1 \rightarrow -\overline{A_1C_1} \cdot s(\theta) - \overline{C_1B_1} \cdot c(\gamma_1) - z = 0 \quad (23)$$

$$i_2 \rightarrow -\overline{C_2B_2} \cdot s(\gamma_2) \cdot c(\alpha_2) - x = 0 \quad (24)$$

$$j_2 \rightarrow -\overline{OA_2} + \overline{A_2C_2} \cdot c(\theta) + \overline{C_2B_2} \cdot s(\gamma_2) \cdot s(\alpha_2) + \overline{B_3P} + y = 0 \quad (25)$$

$$k_2 \rightarrow -\overline{A_2C_2} \cdot s(\theta) - \overline{C_2B_2} \cdot \mathrm{c}(\gamma_2) - z = 0 \quad (26)$$

$$i_3 \rightarrow \overline{OA_3} \cdot \mathrm{c}(30) - \overline{A_3C_3} \cdot \mathrm{c}(\theta) \cdot \mathrm{c}(30) - \overline{C_3B_3} \cdot \mathrm{s}(\gamma_3) \cdot \mathrm{c}(30 + \alpha_3) - \overline{B_3P} \cdot \mathrm{c}(30) + x = 0 \quad (27)$$

$$j_3 \rightarrow -\overline{OA_3} \cdot s(30) + \overline{A_3C_3} \cdot c(\theta) \cdot s(30) + \overline{C_3B_3} \cdot s(\gamma_3) \cdot s(30 + \alpha_3) + \overline{B_3P} \cdot s(30) - y = 0 \quad (28)$$

$$k_3 \rightarrow -\overline{A_3C_3} \cdot s(\theta) - \overline{C_3B_3} \cdot c(\gamma_3) - z = 0 \quad (29)$$

,where $\overline{OP}$ is vector from the point O(x, y, z) to the point P(u,v,w),
$\overline{OA_i}$ is vector from the point O(x, y, z) to the point A_i,
$\overline{A_iC_i}$ is vector from the point A_i to the point C_i,
$\overline{C_iB_i}$ is vector from the point C_i to the point B_i,
$\overline{B_iP}$ is vector from point B_i to the point P(u,v,w),
s is sine,
c is cosine.

Equations (21)–(29) describe the x, y and z components of the vector loop for each limb. In this type of delta robot, which uses prismatic actuators, the only parameters that can be controlled are the three linear screw rail positions. In inverse kinematics the goal is to find the correct joint values that produce the desired end effector position. This means that it would be ideal to solve those 9 equations presented by Uyar and Mutlu (2012) even further so that the linear screw positions would be presented as a function of the x,y,z position of the moving platform. Equations (34)−(42) in appendix A show how the unknown variables α_i and γ_i (i = 1, 2, 3) are solved from equations (21)−(29) above.

5 CONTROLLING DELTA ROBOT USING TWINCAT 3 SOFTWARE

In this chapter the prototype of a linear actuated delta robot is introduced, which is later referred to as JOTDelta. The main principles of the mechanical structure are discussed and the Beckhoff´s TwinCAT 3 software based controlling system of the robot is reviewed. This chapter shows how the inverse kinematics solution of the delta robot, which was found in the previous chapter, can be utilized in the control logic of the JOTDelta. Also the PLC program that is used for the accuracy measurement later in chapter five is discussed. The JOTDelta was designed and manufactured formerly during an internship period in summer 2019 by the author of this thesis, Mika Muurinen and Henri Remes.

5.1 JOTDelta

The JOTDelta is assembled in an aluminum profile frame, which is presented in figure 7. A triangular pyramid shaped rigid structure is mounted in the upper part of the aluminum profile frame. To each side of the triangular pyramid the linear screw rails, which are actuated by servo motors, can be placed securely. Each screw axis has a maximum stroke of 120 mm and can therefore adopt any position between zero and 120 mm. All three screw rails are attached to the moving platform by two aluminum rods. In both ends of the aluminum rods there is a spherical joint. Like usually in the case of a delta robot, only one joint is actuated and the spherical joints are passive. This structure forms a 3-PSS kinematic chain.

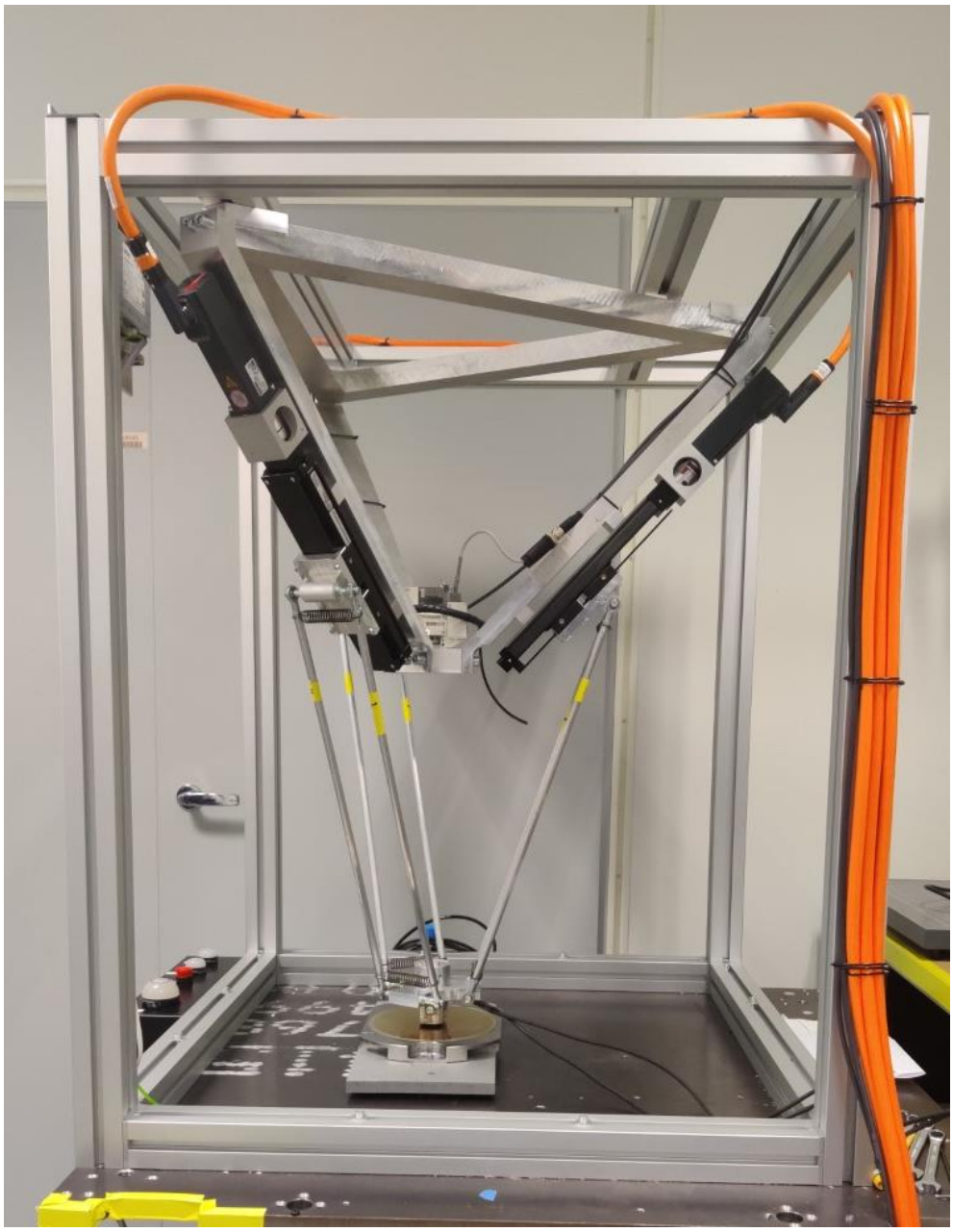

Figure 6. The JOTDelta.

The end effector of the JOTDelta moves in Cartesian space along the virtual x-, y-, and z-axis. As shown in figure 5 the world coordinate system O(x, y, z) is located on top of the JOTDelta. When programming the robot it would be reasonable that the origin is in the center of the work space. By doing this arrangement, the end effector would be in position (0, 0, 0) when all three screw axes are in the middle of their maximum stroke at 60 mm. The translation of the world coordinate system O(x, y, z) can be done by

modifying the k_i $(i = 1,2,3)$ component in the vector loop equations (23), (26) and (29) presented in chapter 3. By an iterative method it was seen that by adding a value 541.3 to each three equations (23), (26) and (29) the origin was set to the desired height.

Figure 8 illustrates the situation after the world coordinate system O(x, y, z) is moved by 541.3 mm. When all three linear screw rails are in the middle of their stroke at position 60 mm, the world coordinate system O(x,y,z) and the moving coordinate system P(u,v,w) are coincident. Now the moving coordinate system P(u,v,w) can be driven in the world coordinate system O(x,y,z) by moving the three linear screw rails, which are notated as Axis 1, Axis 2 and Axis 3 in figure 8.

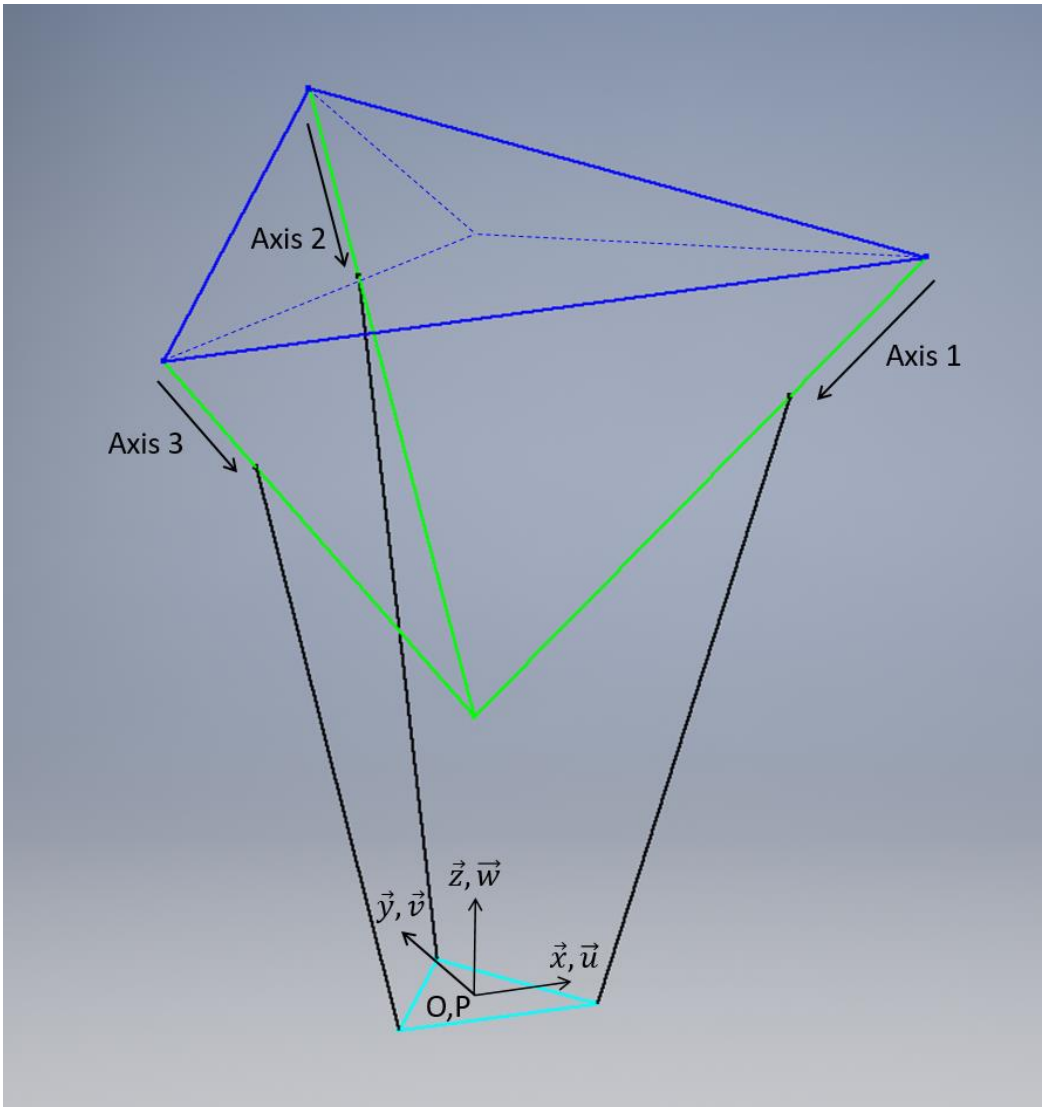

Figure 7. World coordinate system O(x,y,z) and moving coordinate system P(u,v,w).

The vector loop equations (21)–(29) presented in the chapter 3 are universal and can therefore be applied to any delta robot with corresponding geometry. The JOTDelta can be scaled to the desired size for different applications and still the same equations can be used. The length of the vectors presented in the vector loop equations (21) – (29) were obtained from the 3D model of the robot for every (i = 1, 2, 3) as shown in the equations (30)–(33).

$$OA_i = 281 \text{ mm}, \tag{30}$$

$$A_iC_i = 0 \text{ mm} - 120 \text{ mm}, \tag{31}$$

$$C_iB_i = 532 \text{ mm}, \tag{32}$$

$$B_iP = 50 \text{ mm}, \tag{33}$$

Where OA_i is the distance from the world coordinate system O(x, y, z) to the upper end of the rods at screw axis position 0. Vector A_iC_i is the length of the linear screw axis and can adopt values between 0 mm and 120 mm. C_iB_i is the length between the spherical joints on both ends of the rods. Vector B_iP is the length from the lower spherical joint of each rod to the center point of the moving platform P (u, v, w) and in this case has a value of 50 mm.

5.2 TwinCAT 3 software

Beckhoff´s TwinCAT 3 software is an automation software, which can be used in various control applications such as manufacturing processes, construction machines or in public buildings. The TwinCAT 3 uses Visual Studio by Microsoft as a software development environment. The TwinCAT 3 software supports multiple PLC programming languages that are defined in the IEC 61131-3 standard, but can also be programmed using C and C++ or MATLAB/Simulink. In this case the PLC is programmed using the ST language (Structured text) (Beckhoff, 2020b).

There is also a motion control package included in the TwinCAT 3 software. This eXtended Motion Control feature offers solutions to point-to-point motion, NCI (Numerical Control Interpolation), CNC (Computerized Numerical Control) and robot control (Beckhoff, 2020a). Point-to-point movements are usable when only the starting point and the end point of the movement matters. This means that the end effector can move from the starting position to the end position by any possible route between these points. In some cases, such as machining or in robotics, we are interested also in the path of the movement. In applications like milling or robotic assembly it is crucial that not only the starting point and the end point but also the path is as described. In these applications the NCI feature can be used. The eXtended Motion Control offers interpolated movements with three axes and five additional axes.

Also interpolated motion for robotic control is supported. This feature includes the kinematic transformations for several different mechanisms. It can be used to calculate the kinematics for different mechanisms and the motion control can be formed efficiently. The kinematic transformations, which include forward transformation, inverse transformation and dynamic modelling are supported for different 2D mechanism, the SCARA robot and the delta robot. In the kinematic transformation for the delta robot the inner and outer arm lengths are parametrized. Also the displacement from the center to the rotary actuators can be set.

The major downside of eXtended Motion Control kinematic transformation package is that only the delta robot with rotational actuators is supported. In this case the JOTDelta is actuated by linear screw rails and the kinematic package cannot be used.

5.3 Programming the JOTDelta

In this chapter the program that is used to make NCI movements with the JOTDelta is introduced. Also the usage of inverse kinematics equations (36), (39) and (42) shown in appendix A are discussed.

5.3.1 State machine based NCI program

The program that is being used in controlling the JOTDelta is based on an NCI example program, which can be downloaded from Beckhoff´s website (Beckhoff 2020c). This example program is only to demonstrate how to start a program that uses interpolated movements so it needs modifications to be able to run the JOTDelta. The program begins by reading the status of every axis. In this program there are virtual axes x, y and z that demonstrate the Cartesian coordinate system. The first thing to do is to add three more axes so the actual screw rails can be operated. After this the axes are powered by using the MC_Power function block. This function sets the bEnable to true, which allows the axes to be driven.

The program is based on a state machine. All possible states are shown as circled in figure 9. Also the conditions for the state to change are shown next to the arrows in figure 9. The state machine can be used to divide the program into reasonable pieces and every state is responsible for one particular feature.

When the JOTDelta is switched on, the state machine begins at state zero, which in this case is called Init. At this state the program just waits that the bExecute is set to true, which means that the user has pressed the start button and wants the robot to actually operate.

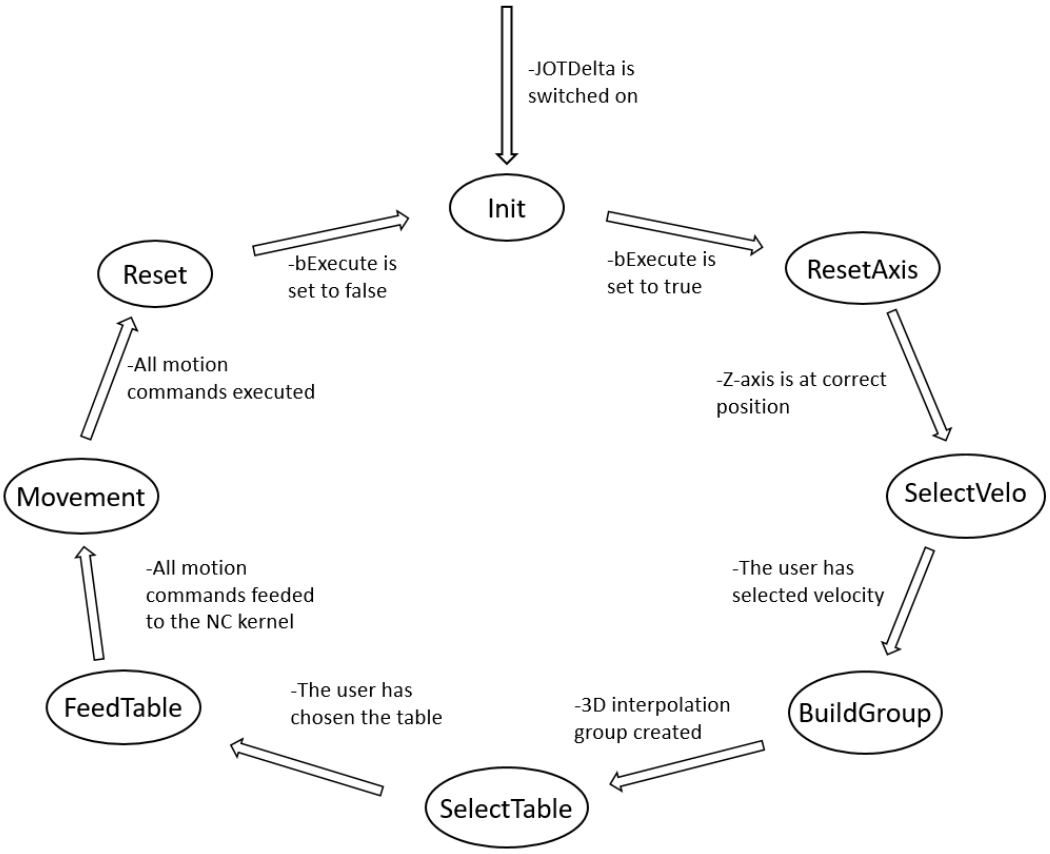

Figure 9. All possible states and conditions of the state machine.

After the bExecute is set to true the state machine moves to the next state, which is ResetAxis. This state is used to set the z-axis to the desired height at the beginning of the program. When the program is started the position of each screw rail can be read from the encoders of the servo drives. However, the virtual x-, y- and z-axes are also used in controlling the JOTDelta and these axes do not have encoders. Because of the lack of encoders in virtual axes, these axes are at position zero at the beginning of the program.

After this the NciState is set to SelectVelocity in which the user can set the velocity of the end effector as needed. When the correct velocity is accepted by the user the state machine moves to next state, which is called BuildGroup. In this step the CfgBuildExt3DGroup function from the PLC NCI library is called. This function creates a 3D interpolation group from the virtual x, y and z-axes. By using this function it is now possible to create interpolated path movements with the virtual x, y and z-axes. After the interpolation group is created the state machine moves to next state SelectTable.

In state SelectTable it is possible to choose which table is to be used in the movement. The tables are created under the main program as actions. In these actions the actual moving commands are written with structured text. All possible motion commands, which are type Struct, are found from the PLC library TcPlcInterpolation. This library offers an alternative of using G-code and interpolated motion commands are run directly from the PLC. Every action begins by calling the function block FB_NciFeedTablePreparation, which appends a table of NCI movements to the PLC. This function block takes as an input movement commands, such as starting point of the movement, straight line command or circle command. After all motion commands are filled in the table the state machine moves to next state, which is FeedTable.

In the state FeedTable the function block FB_NciFeedTable is called. This function block takes previously created motion commands as an input and transfers them to the NC kernel (Numerical Control), which executes the motion commands. After all motion commands are feeded to the NC kernel the state machine moves to state Movement.

In the Movement state all three linear screw axes are driven. At first the LinAxisPos function is called. This function contains the inverse kinematic functions to all three screw axes. The inverse kinematics functions take as an input the x, y, and z position of the virtual axes.

The actual position of the virtual axes that are being interpolated can be read from the Tc2_MC2 library under AXIS_REF. These actual positions of the interpolated x, y and z axes are fed to the inverse kinematics functions as an input and the screw axis positions that correspond that point can be calculated.

After the new target position for every screw axis is calculated the function block MC_ExtSetPointGenEnable from the Tc2_MC2 library is called. This function switches the external set point generator of each axis on and the axis adopts the new set point shown in figure 10. After this the function block MC_ExtSetPointGenFeed is called and the screw axes are driven. The MC_ExtSetPointGenFeed function block takes as an input the target position calculated previously in the inverse kinematics functions. Also the velocity that was set by the user in state SelectVelocity is taken as an input.

```
7  MC_ExtSetPointGenFeed(
8          Axis := _Axis^,
9          Position := Position,
10         Velocity := Velocity,
11         Acceleration := 2500,
12         Direction := 1);
```

Figure 10. Driving the screw rails in the PLC program.

In every PLC cycle new x, y, z target position from the interpolated virtual axes is generated, corresponding screw axis positions are calculated and the screw axes are driven. Once the all motion commands are executed the bChannelDone bit of the function block FB_NciFeedTable is set true and the bExecute is set as false. The state machine can now move to Reset state where the program ensures that the bExecute is truly false. After this the state machine moves to state zero.

5.3.2 Adding the inverse kinematics equations to the PLC program

The inverse kinematics equations (36), (39) and (42) for all three linear screw axes are shown in appendix A. Each of these three equations A_iC_i describe the screw axis position as a function of x, y, z and A_iC_i position. To ease the inverse kinematics calculation for the PLC and the programming phase it is reasonable to modify the equations so that the equations take in only the target position xyz-coordinate as three variables.

This modification can be done easily by using the MATLAB software. The MATLAB script, which is used to solve the unknown screw axis position A_iC_i as a function of x, y and z position variables is shown in appendix B. The script begins by defining the variables OA_i, C_iB_i, B_iP that are being used in the program. Also the layout angle variables θ_i (i = 1, 2, 3) of the screw axes are defined and the z_offset variable, which moves the origin O(x, y, z) to the desired height is placed. After this the inverse kinematics equations (36), (39) and (42) are written and stated as eqn_i (i = 1, 2, 3).

After this the equations can be solved using the MATLAB function solve(eqn, var). The solve(eqn, var) function takes as an input an equation and a variable that is being solved.

In this case the solve function is used to solve equations eqn_i for the variables A_iC_i. By adding a MATLAB function vpa(x, d) the script evaluates a symbolic input x to at least d digits. In this case the vpa(x, d) function is used to evaluate eqn_i functions to at least 10 digits, which should give accurate enough solution to control the JOTDelta.

The MATLAB script solves the three inverse kinematic equations eqn_i as a function of x, y and z position. These solutions are now in correct form to be used in the PLC program to control the JOTDelta. The three final equations can be placed to the LinAxisPos function in the PLC program, which was discussed in the previous chapter.

6 ACCURACY MEASUREMENTS OF THE JOTDELTA

This chapter focuses on accuracy measurements of the JOTDelta. The preparations that are needed for successful accuracy measurements are discussed. Also the measurement process is presented.

6.1 Measurement preparation

The accuracy measurement preparation was done during the internship period in summer 2019 by the author of this thesis, Mika Muurinen and Henri Remes. During the first test drives with the JOTDelta it was seen that the programmed movements were not that accurate. The program contained straight line movement commands and still the motion of the end effector was more like a parabola. The inaccuracy of the test drive paths was confirmed by attaching a dial indicator to the moving platform.

This issue was solved by using the Absolute Arm 6-axis by Hexagon. The Absolute Arm is a portable 6-axis measuring arm, which can be used in high accuracy measurements. Each 6 axes of the Absolute Arm are equipped with an absolute encoder. In the tip of the Absolute Arm is a touch probe and whenever the user presses the recording button the absolute position of the touch probe is recorded according to the position information of the encoders. This measuring arm can be used to measure distances, angles and planes.

The Absolute Arm was used to verify the crucial measures of the JOTDelta. As discussed earlier discussed in chapter 2.4, the differences between the nominal link length values that are being used in the kinematic equations and the actual link lengths can cause inaccuracy to the robots movements. The dimensions of the JOTDelta's structure that were used in the inverse kinematics equations were obtained from the 3D model of the robot. The JOTDelta is built up from different machined parts and it was obvious that there is some slight error in dimensions that causes the inaccuracy of the robot. By measuring all crucial dimensions of the JOTDelta the inverse kinematics equations in the control logic can be edited so that they correspond to the actual dimensions of the robot. The measurement process begins by mounting the Absolute Arm on a stable surface. After this the touch probe is placed on the surface being measured and the user can save

measurement points of the touch probe by pressing a button on the side of the Absolute Arm.

The dimensions of all six rods were studied by measuring the distance from the upper spherical joint to the lower one. Also the layout angle of the screw axes was measured by creating two planes from a set of data points measured from the upper triangle shaped frame and the mounting beams for the screw axes.

6.2 Accuracy measurement equipment

The accuracy measurement of the JOTDelta was performed using the Heidenhain KGM182 grid encoder. This grid encoder is used to evaluate the accuracy of machine tools. The grid encoder can be used to measure dynamic behavior and the positioning accuracy of the machine tool in different use cases. By these measurements the control loop of the machine tool can be studied and also the accuracy of the position feedback is checked. Also the effect of ambient temperature can be verified. The same equipment can be used to evaluate the positioning accuracy of the JOTDelta's XY-plane.

The equipment needed for this accuracy measurement is the KGM182 grid encoder with the scanning head, EIB741 External Interface Box and a PC with the ACCOM software, which is shown in figure 11. At the beginning of the measurement process the scanning head is mounted to the machine, which is being inspected and the grid encoder is secured on the table. During the measurement the scanning head moves over the grid encoder. The EIB741 has four encoder inputs in total (X11-X14). In this measurement only two encoder inputs are used. The encoder input X11 reads the x-axis data while the encoder input X12 reads the y-axis data. The EIB741 is connected to the PC with a standard Ethernet interface for data output.

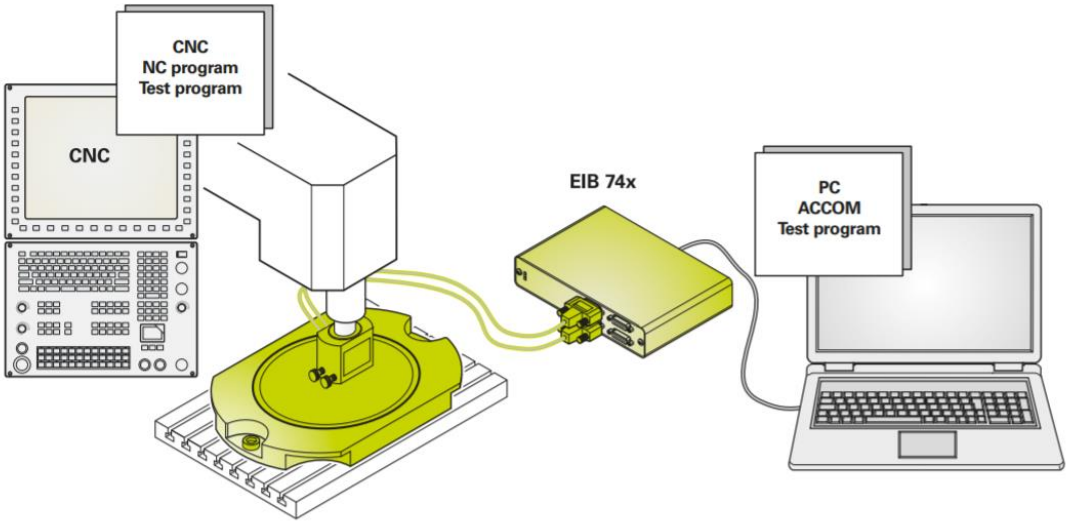

Figure 11. Measurement setup (Heidenhain, 2017).

During the measurement the scanning head moves over the grid encoder without making mechanical contact. The gap between the scanning head the grid encoder is kept at 0.5±0.05 mm. The grid encoder has a geometric pattern, which consist of 2.83 µm sized squares shown in figure 12. The scanning head has to be perfectly lined with the grid encoder. This is ensured by turning the adjustment screws in the scanning head.

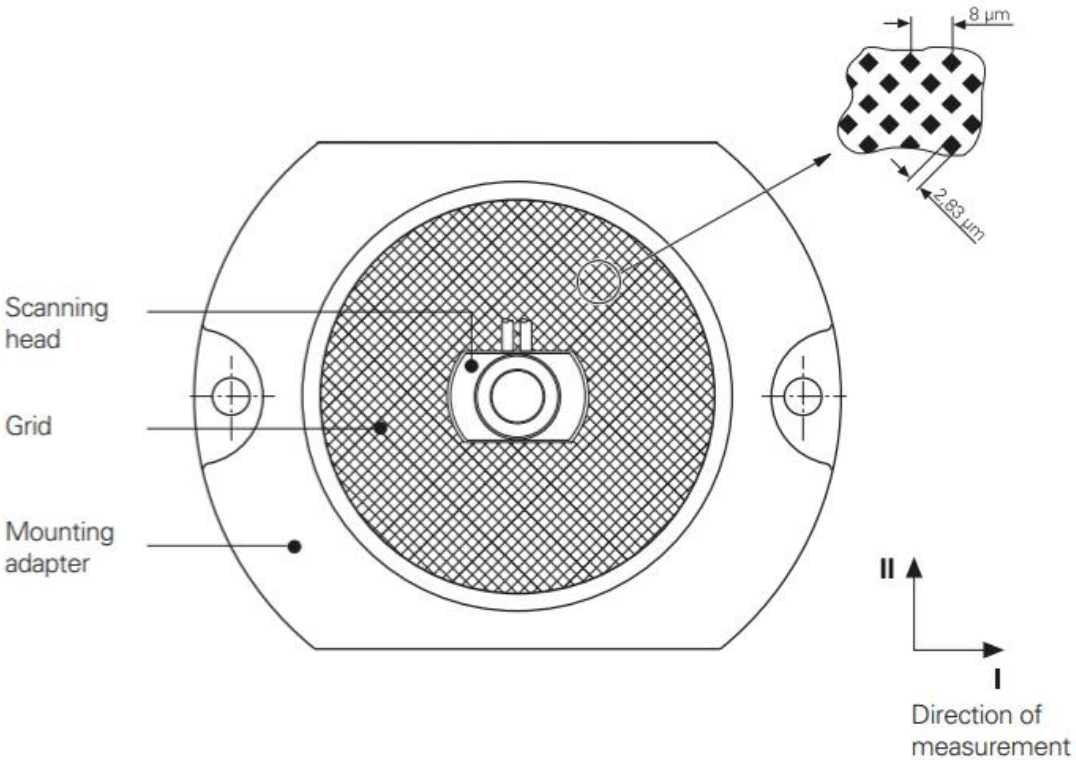

Figure 12. The KGM182 grid encoder and the scanning head (Heidenhain, 2017).

After the measurement setup is ready the ACCOM 3.2 software can be opened. At first the program asks to select the measuring method. In this measurement the KGM free-form test was used By using the KGM free-form test as measurement method, more complex drive paths can be studied.

After this the units of the measurement are set to (mm). Also the signal period is set to match the encoder type being used. In this case the signal period is the default value 4 µm. The next thing to do is to select the counter card, which in this measurement was the EIB741 Also the IP address of the EIB741 can be set here. Alternatively the IP address of the PC, which is running the ACCOM software can be changed to correspond with the EIB741 IP address.

The next thing to do is to ensure that the scanning head is perfectly lined with the grid encoder. This can be done by selecting Equipment → Oscilloscope in the ACCOM 3.2 software. This opens the oscilloscope view, which is used to set the adjustment screws in

the scanning head to a correct position. The oscilloscope view shows the encoder signals X11 and X12 that are received in the EIB741. The target is to set the encoder signals, which are shown as black circles in figure 13, between the two red circles. When the encoder signals are right between the two red circles, on the green circle, the scanning head is lined with the grid encoder. If the gap between the scanning head and the grid encoder is too big the encoder signal cannot be adjusted to the correct level even with the adjustment screws.

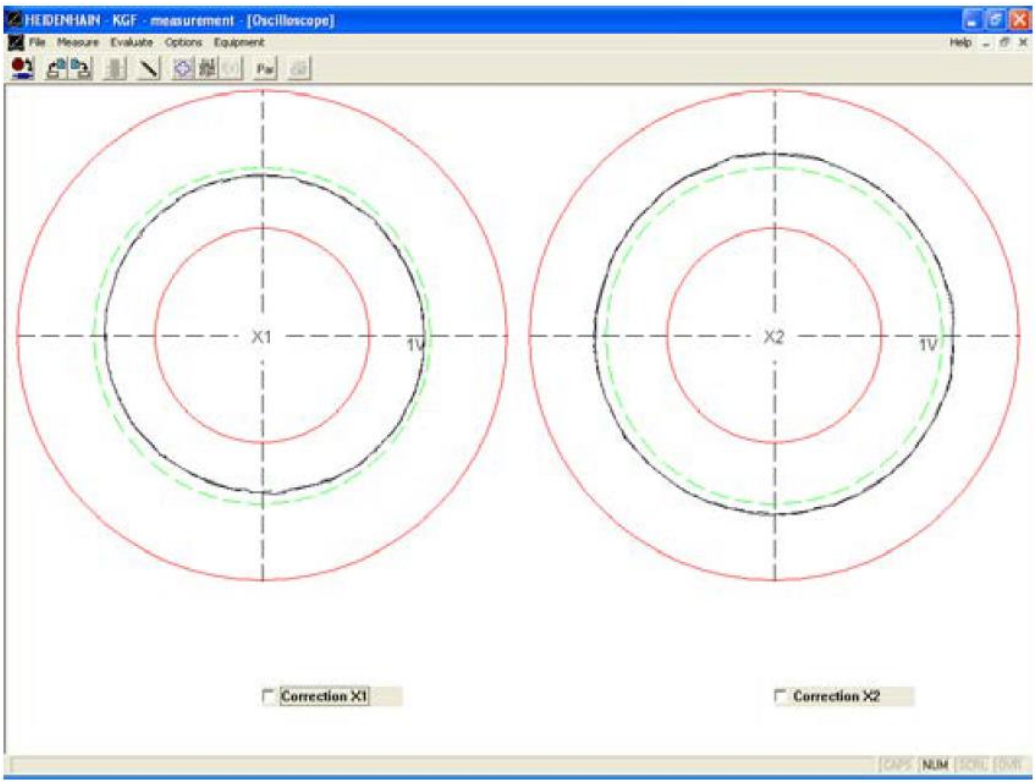

Figure 13. Adjusting the signal of the scanning head.

The next thing to do is to set the directions of the measurement as desired. In this point the scanning head is driven to the positive x-direction. After this, also the positive y-direction is taught. By doing this the ACCOM knows the positive direction of both axes but also the angular position of the grid encoder on the table.

The path of the free-form test can be entered by selecting Measure → Enter path. Here the user can define the path being used in the measurement. The path, which is to be used

in the accuracy measurements for the JOTDelta, consists of straight line movements, which form a star shaped pattern with a diameter of 120 mm shown in figure 14. The measurement path begins from the origin. Next the robot is programmed to move 60 mm strokes to the arc of the pattern. Before heading to the next point on the arc of the star pattern, the robot returns to the origin. There are in total 16 points on the arc of the pattern which are gone through.

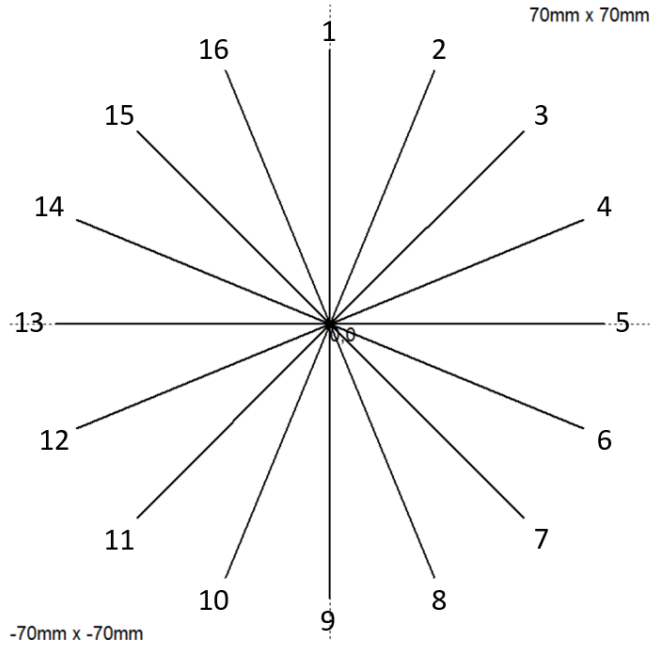

Figure 14. Measurement path.

When defining the measurement path the first thing to do is to set the feed rate and the starting point. After this, the actual movement commands are filled in by entering a straight line command and entering the target XY-position. A few example commands are shown in the figure 15.

```
Feed rate     600 mm/min
Starting point X0 mm    Y0 mm
Straight line  X0 mm    Y60 mm
Straight line  X0 mm    Y0 mm
Straight line  X22.961000 mm    Y55.4328 mm
Straight line  X0 mm    Y0 mm
```

Figure 15. Path description in the measurement program.

Figure 16 shows the measurement arrangement. The figure shows the Heidenhain KGM182 grid encoder and the scanning head. The scanning head was mounted securely to the moving platform by using a screw joint.

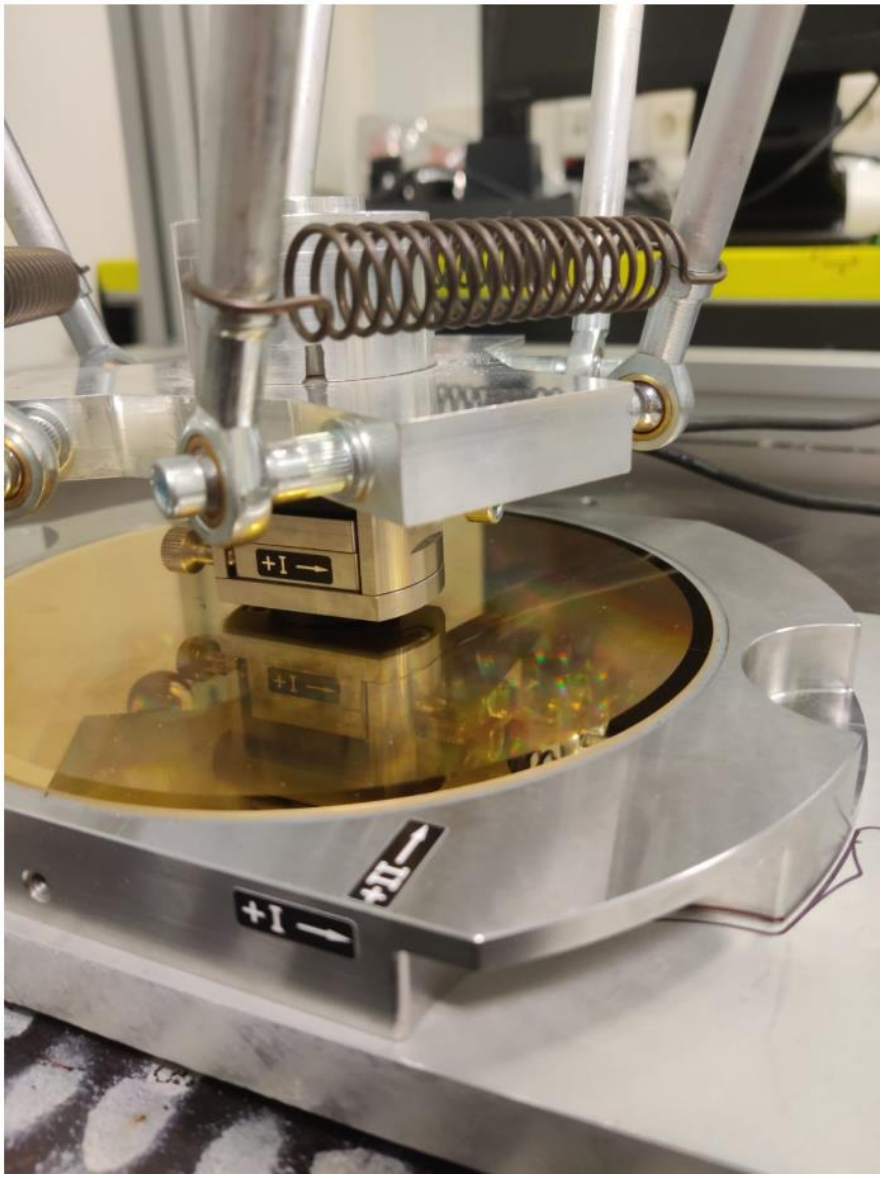

Figure 16. The KGM 182 grid encoder and the scanning head.

7 RESULTS AND DISCUSSION

In this chapter the results of the positioning accuracy and repeatability measurements of the JOTDelta are discussed. Also the positioning accuracy of ABB's Flexpicker IRB 360-1/1130 delta robot is shown and a small-scale comparison between the results is made.

7.1 JOTDelta results

The positioning accuracy and repeatability of the JOTDelta was measured by driving a star shaped pattern with different velocities. From the measurement data shown in this chapter the accuracy and the repeatability of the JOTDelta can be evaluated. The same star shaped pattern with a diameter of 120 mm was measured 40 times with velocities of 10 mm/s, 50 mm/s, 200 mm/s, 300 mm/s and 400 mm/s.

After the measurement, the measurement data can be evaluated graphically in the ACCOM software as shown in figures 17 and 18. The programmed target path is marked with a black line and the actual path of the end effector, which was measured with the KGM182 grid encoder, can be seen as a red line.

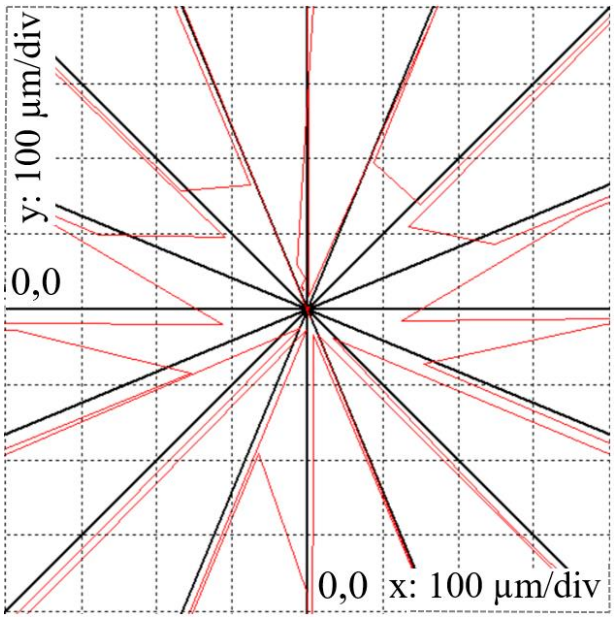

Figure 17. Measurement data around the origin using 10 mm/s velocity.

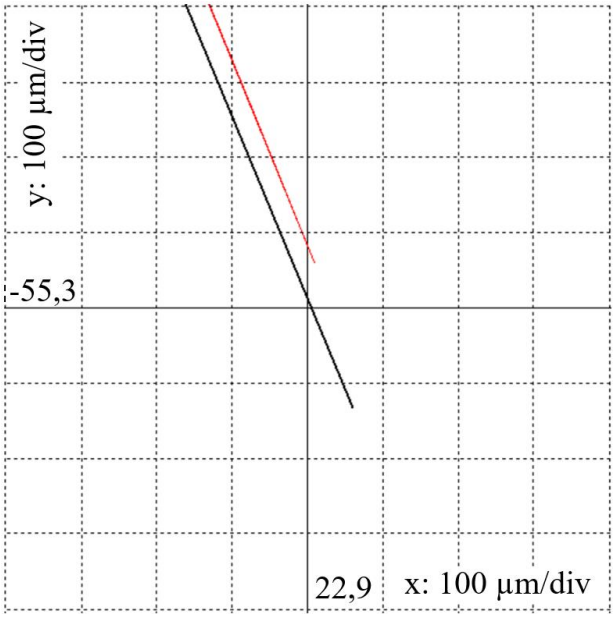

Figure 18. Measurement data around the point 8 using 10 mm/s velocity.

The XY-position of the scanning head was measured 4000 times in each measurement. Measurement data can be imported to Microsoft Excel software for efficient analyzing. Table 1 presents the positioning inaccuracy of the JOTDelta in target points 1−16 for all velocities. The values are shown in micrometers (µm) and calculated by using equation (4).

Table 1. Positioning inaccuracy of the JOTDelta in micrometers (µm).

	10 mm/s	50 mm/s	200 mm/s	300 mm/s	400 mm/s	Average
Point 1	301.9	235.9	257.2	259.5	216.5	254.2
Point 2	339.2	180.0	226.6	244.5	182.2	234.5
Point 3	312.3	122.5	106.9	117.3	96.9	151.2
Point 4	301.3	148.1	104.2	142.4	88.1	156.8
Point 5	280.4	130.5	52.1	65.3	104.5	126.5
Point 6	243.2	83.7	38.4	51.4	37.8	90.9
Point 7	168.3	43.1	14.3	3.1	16.6	49.1
Point 8	163.1	9.4	1.4	24.1	3.9	40.4
Point 9	225.6	115.5	103.5	217.3	201.2	172.6
Point 10	274.6	155.3	148.1	213.3	182.8	194.8
Point 11	330.8	169.5	134.9	136.1	135.1	181.3
Point 12	355.5	205.2	170.0	217.2	169.4	223.4
Point 13	324.8	170.1	121.9	134.7	156.6	181.6
Point 14	230.2	83.4	62.0	78.1	54.1	101.6
Point 15	204.2	51.8	33.2	34.7	32.4	71.3
Point 16	251.5	125.5	128.9	141.2	92.2	147.9

By using the measurement data of table 1 the positioning accuracy can be presented as a dotted line graph for better evaluation as shown in figure 19. The positioning accuracy was measured only in the target points, which are presented as dots. The dashed line between the dots is only for better visualization and does not present any measurement.

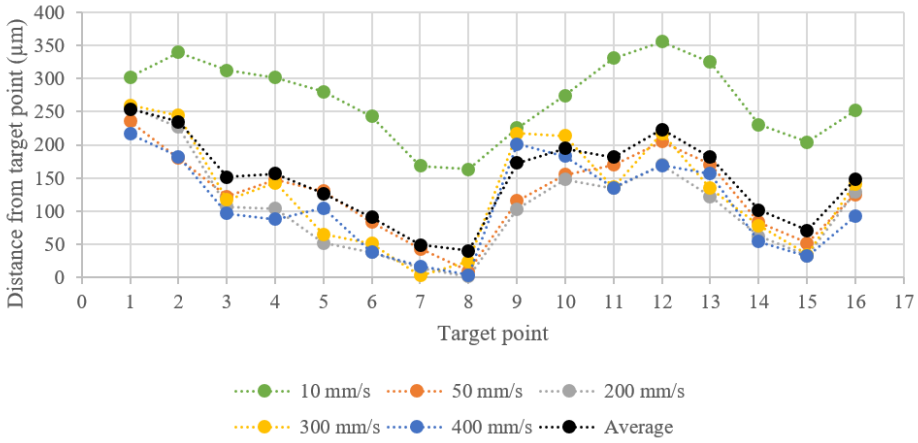

Figure 19. Positioning accuracy of the JOTDelta.

From figure 19 it can be seen that the positioning accuracy of the JOTDelta varies a lot depending on the target position. The vertical axis shows the distance from the target point in µm, while the horizontal axis describes the target points. Perhaps the most interesting information in figure 19 is that the weakest result for the positioning accuracy was obtained with the lowest velocity of 10 mm/s. Generally said, the best results should be obtained with the lowest velocity and vice versa. This might be traced to the PLC program of the JOTDelta. All the other velocities produced almost the same results so it is interesting why the JOTDelta reacts like this with only the velocity of 10 mm/s.

From the average value, which is shown as a black line in figure 19, it can be noticed that the point number 8 is the most accurate point and has an error value of 40.4 µm. Also from the points 7 and 15 quite good results were obtained, with values of 49.1 µm and 71.3 µm. The poorest result in average is in the first target point, with a value of 254.2 µm.

The position repeatability was calculated by using equation (8). The position repeatability of the JOTDelta is shown in the table 2. Also the positioning repeatability was calculated in target points 1-16 for all velocities.

Table 2. Positioning repeatability of the JOTDelta in micrometers (µm).

	10 mm/s	50 mm/s	200 mm/s	300 mm/s	400 mm/s	Average
Point 1	182.2	199.1	389.8	311.7	213.0	259.2
Point 2	183.5	49.3	206.2	181.0	123.1	148.6
Point 3	132.1	55.0	111.7	111.4	78.0	97.7
Point 4	160.2	41.4	64.1	69.0	50.7	77.1
Point 5	179.8	33.6	36.1	40.6	30.1	64.0
Point 6	178.9	44.1	19.3	25.7	18.8	57.4
Point 7	133.1	60.0	31.5	55.6	40.0	64.0
Point 8	161.7	33.9	21.5	15.1	24.0	51.2
Point 9	172.7	147.8	297.0	170.2	442.3	246.0
Point 10	169.1	42.7	164.0	98.3	259.8	146.8
Point 11	118.7	68.1	88.0	62.0	158.4	99.1
Point 12	172.5	47.2	49.1	35.0	102.3	81.2
Point 13	168.4	35.8	29.6	21.1	56.7	62.3
Point 14	175.6	40.5	19.9	14.6	32.1	56.6
Point 15	117.4	56.0	40.0	35.6	36.0	57.0
Point 16	180.3	44.9	31.4	19.8	16.3	58.5

The data from the table 2 is presented as a dotted line graph in figure 20. Only the dots present the actual measured values and the dashed line is for visualization. The vertical axis presents the position repeatability in µm and horizontal axis presents the different target points.

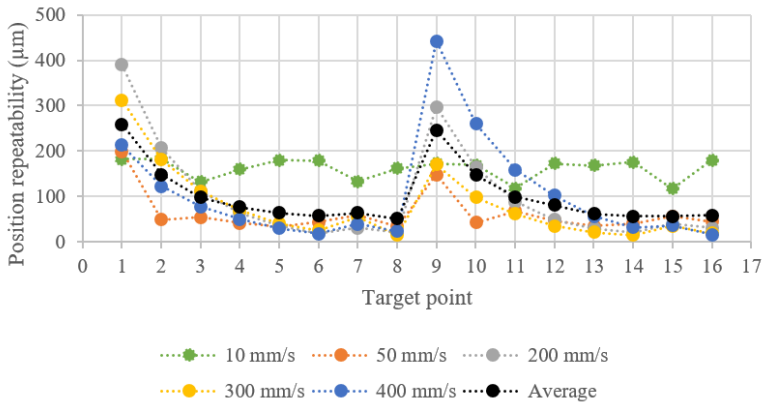

Figure 20. Positioning repeatability of the JOTDelta.

The first thing to notice from figure 20 is that the weakest results are from the points 1 and 9. This might also be traced to the PLC program. In the program the motion commands are feeded to the NC kernel in two pieces. By splitting the program in two pieces, the JOTDelta drives through first the points 1−8. After these movements are completed the other half of the target points are feeded into the NC kernel and drive commands to points 9−16 are executed. It seems that the weakest repeatability is attained in the first target points (1. and 9.) of both tables. After this the repeatability improves towards the last target points (8. and 16.) in both tables.

One interesting thing is also that for some reason the measurements with the velocity of 10 mm/s acts again totally different than the others. The repeatability stays below 200 µm in every target point and no such peaks are shown in points 1 and 9 like with the other velocities. The position repeatability stays below 450 µm in every point and every velocity.

7.2 Results comparison

ABB's Flexpicker IRB 360-1/1130 delta robot, which is shown in figure 21, (later referred to as the Flexpicker) was tested by using the same measurement equipment. Unfortunately only one measurement with each velocity was done with the Flexpicker and because of this the results are not fully comparable.

Figure 21. ABB´s IRB 360 Flexpicker (ABB, 2017).

Table 3 shows the positioning accuracy of the Flexpicker in points 1−16 and for all velocities 10– 400 mm/s. The information of the table is also presented in figure 22 with dotted lines. It is highly noticeable that the line graph in figure 22 is based on only one measurement for each velocity. The measured points are presented as dots and the dashed lines are only to ease interpreting the figure.

Table 3. Positioning inaccuracy of the Flexpicker in micrometers (µm).

	10 mm/s	50 mm/s	200 mm/s	300 mm/s	400 mm/s	Average
Point 1	44.2	43.5	71.2	48.2	4.1	42.2
Point 2	89.3	76.1	75.6	77.4	57.0	75.1
Point 3	67.6	55.3	54.9	67.2	45.1	58.0
Point 4	90.0	85.7	115.7	62.3	64.0	83.6
Point 5	80.7	98.4	116.8	57.3	52.8	81.2
Point 6	89.2	86.7	97.8	68.5	36.7	75.8
Point 7	81.3	96.0	82.2	64.1	64.8	77.7
Point 8	79.0	91.6	86.0	70.8	75.1	80.5
Point 9	86.8	72.7	56.6	35.4	38.7	58.0
Point 10	57.6	57.1	88.4	53.4	43.4	60.0
Point 11	80.7	60.3	95.6	34.7	34.8	61.2
Point 12	22.0	14.3	1.4	7.1	8.7	10.7
Point 13	84.9	86.1	72.1	58.1	63.2	72.9
Point 14	88.3	88.6	77.5	57.7	22.5	66.9
Point 15	119.0	136.4	115.0	111.5	81.7	112.7
Point 16	97.7	101.0	94.6	95.0	85.8	94.8

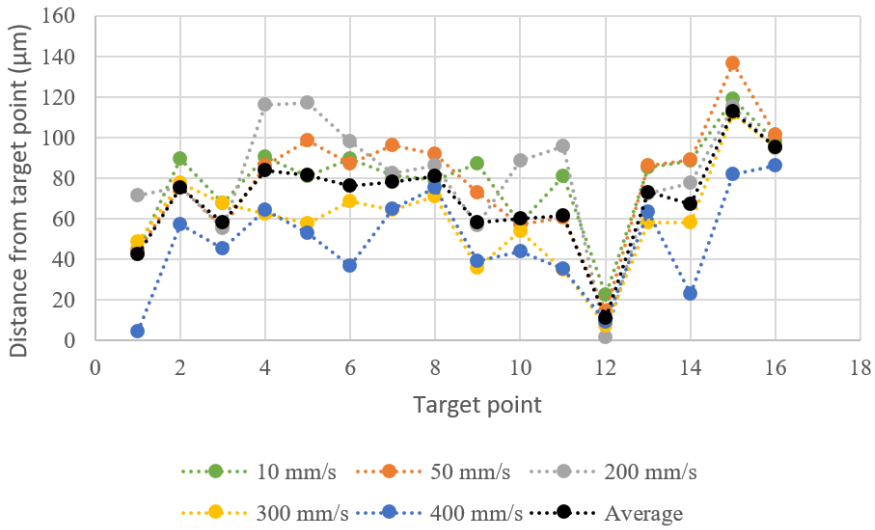

Figure 22. Positioning inaccuracy of the Flexpicker.

From figure 22 it can be seen that the positioning inaccuracy stays under 140 µm in every point and every velocity. Also noticeable is that point 12. is the most accurate, and the maximum error is only 22 µm. The repeatability of the Flexpicker could not be evaluated due to lack of measurements. ABB informs in the datasheet of the Flexpicker that the position repeatability is 100 µm. In comparison to the JOTDelta the positioning accuracy and the repeatability of the Flexpicker seems better. However, it is noticeable that the repeatability of the Flexpicker is from the ABB's datasheet and, therefore, can be biased information.

8 SUMMARY

This thesis started with a review of the industrial robot features and the markets related to them. The most common industrial robot types were presented and compared. After this the forward and inverse kinematics problem of an open chain mechanism were studied.

The kinematics discussion continued in chapter four, which focused on the delta robots. Different methods to solving the forward kinematics problem were presented from the literature and the inverse kinematics solution of the delta robot was shown. After this the state machine based control logic of a case study delta robot was presented. It was shown how the inverse kinematics solution can be implemented in the PLC program.

The validity of the delta robot's inverse kinematics model was examined by executing a set of tests to a case study delta robot. These tests included measuring the positioning accuracy and repeatability of the robot. The positioning accuracy and repeatability was measured with a grid encoder. Also a small-scale comparison between the case study delta robot and a commercial delta robot was arranged. It was predicted that the commercial delta robot would attain better results in the positioning accuracy and repeatability tests.

The results show that the inverse kinematics model of the delta robot does actually work and good results from the positioning accuracy and repeatability were achieved. The performance of the case study delta robot could not be fully compared to the commercial delta robot.

For the further studies it would be reasonable to start from modifying the PLC program. From the positioning accuracy measurements is was seen that the poorest result was attained with the lowest velocity. It is predicted that this exceptional behavior is due the drive commands from the PLC. It seems that feeding the drive commands from the PLC to the NC kernel causes the most inaccuracy in the results.

These measurements studied only the XY-plane movements of the case study delta robot and it would be wise to measure also the positioning accuracy and repeatability of the z-

axis. By developing the motion control of the case study delta robot further even better results from the positioning accuracy and repeatability would be available. Another aim for further studies is to compare the positioning accuracy and repeatability using different acceleration values in the drive commands.

For further studies it would be a valuable idea to create a calibration tool for the delta robot. Also a feedback system of the end effectors position would make a difference in the results. It would also be interesting to see if the motion control of the delta robot could be fulfilled without using the built-in NCI libraries in the TwinCAT 3 software.

REFERENCES

ABB, 2017. *IRB 360 FlexPicker® Greater flexibility in a compact footprint*. Available at: https://library.e.abb.com/public/021ecff97f374437b2564ee0d7a8aa6f/IRB360_ROB0082EN-H_DataSheet-A4.pdf [14.4.2020].

Altamirano, P., Birkinshaw, M., Chang, C., Chang, S., Chang, S., Chen, K., Chen, M., Chereau, G., Chiueh, T., Han, C., Ho, P.T.P., Huang, C.L., Huang, Y., Ibañez-Romano, F., Jiang, H., Kesteven, M., Koch, P.M., Kubo, D., Li, C., Lancaster, K., Liao, Y., Lin, K., Liu, G., Lo, K.Y., Martin, R.N., Martin-Cocher, P., Molnar, S.M., Nishioka, H., Oshiro, P., Patt, F., Pausch, K., Raffin, P., Romeo, B., Umetsu, K., Wang, F., Wei, T., Willmeroth, K. and Wu, J.P., 2009. *The AMiBA Hexapod Telescope Mount.* doi: 10.1088/0004-637X/694/2/1670. Baqai, A.A., Butt, S.U., Ghazi, M. and Nazir, Q. 2018. *Accuracy Analysis of 3-RSS Delta Parallel Manipulator*. doi: 10.1016/j.promfg.2018.10.033.

Barbieri, L., Bruno, F., Gallo, A., Muzzupappa, M. and Russo, M.L., 2018. *Design, prototyping and testing of a modular small-sized underwater robotic arm controlled through a Master-Slave approach.* doi: 10.1016/j.oceaneng.2018.04.032.

Beckhoff, 2020a. *eXtended Automation*. Available at: https://download.beckhoff.com/download/document/catalog/Beckhoff_TwinCAT3_e.pdf [14.4.2020].

Beckhoff, 2020b. *Getting started*. Available at: https://download.beckhoff.com/download/document/catalog/TwinCAT_3_Booklet.pdf [14.4.2020].

Beckhoff, 2020c. Samples, PlcInterpolationSimpleSample. Available at: https://infosys.beckhoff.com/english.php?content=../content/1033/tf5100_tc3_nc_i/9007202536629003.html&id=4794075640911057467 [9.4.2020].

Bi, Z.M. and Lang, S.Y.T., 2007. *A Framework for CAD- and Sensor-Based Robotic Coating Automation.* doi: 10.1109/TII.2007.891309.

Bird, A., Harrisand, A., Kweder, K., Şirinterlikçi, A. and Tiryakioğlu, M. 2009 *Repeatability and Accuracy of an Industrial Robot: Laboratory Experience for a Design of Experiments Course.* Available at: https://www.researchgate.net/publication/260336817_Repeatability_and_Accuracy_of_an_Industrial_Robot_Laboratory_Experience_for_a_Design_of_Experiments_Course [9.4.2020].

Bonev, I., 2001. *Delta Parallel Robot — the Story of Success.* Available at: http://cats-fs.rpi.edu/~wenj/ECSE641S07/delta-robot_success_story.pdf [4.5.2020]

Chen, H., He, J., Zhang, H., Xi, N. and Zhang, G., 2006. *On-Line Path Generation for Robotic Deburring of Cast Aluminum Wheels*. doi: 10.1109/IROS.2006.281679.

Chen, Y. and Dong, F., 2013. *Robot machining: recent development and future research issues.* doi: 10.1007/s00170-012-4433-4.

Cheng, Y., He, K., Pan, H., Shi, S., Song, Y., Villedieu, E., Yang, Y., and Yang, S., 2017. *Conceptual design of EAST multi-purpose maintenance deployer system.* doi: 10.1016/j.fusengdes.2017.02.042.

Choi, M.H and Lee,W.W,, 2001. *A force/moment sensor for intuitive robot teaching application*. doi: 10.1109/ROBOT.2001.933244

Chong, J.W.S., Nee, A.Y.C., Ong, S.K. and Youcef-Youmi, K., 2009. *Robot programming using augmented reality: An interactive method for planning collision-free paths.* doi: 10.1016/j.rcim.2008.05.002.

Conrad, K.L. and Shiakolas, P.S., 2000 *Robotic calibration issues: Accuracy, repeatability and calibration.* Available at: https://mars.uta.edu/me5337/reference/calibration.pdf [10.4.2020].

Delgado Sobrino, D.R., Holubek, R. and Ružarovský, R., 2018. *Using virtual reality as a support tool for the offline robot programming.* doi: 10.2478/rput-2018-0010

Doan, N.C.N. and Lin, W., 2017. *Optimal robot placement with consideration of redundancy problem for wrist-partitioned 6R articulated robots.* doi: 10.1016/j.rcim.2017.04.007.

Dongsu, W. and Hongbin, G., 2007. *Adaptive Sliding Control of Six-DOF Flight Simulator Motion Platform.* doi: 10.1016/S1000-9361(07)60064-8.

Duysinx, P. and Géradin, M., 2004. *An introduction to robotics: Mechanical aspects.* Available at: https://orbi.uliege.be/bitstream/2268/25611/1/Robotics2004.pdf [15.4.2020]

Ertunç, H.M., Güner, H.E., Özakyoland, H., 2019. *Kinematic Analysis and Design of 3-PUU Parallel Manipulator.* Available at: https://www.researchgate.net/publication/336254050_Kinematic_Analysis_and_Design_of_3-PUU_Parallel_Manipulator [10.4.2020].

Frey, G., Heck, I., Kraus, P. and Stephan, P., 2009. *Evaluation of indoor positioning technologies under industrial application conditions in the Smartfactory^{KL} based on ISO 9283.* Available at: https://www.uni-saarland.de/fileadmin/user_upload/Professoren/FreyG/GF_PS_IH_PK_INCOM_jun_2009.pdf [14.4.2020].

Heidenhain, 2017. *Measuring systems for machine tool inspection and acceptance testing.* Available at: https://www.heidenhain.com/fileadmin/pdb/media/img/208871-29_Measuring_Devices_For_Machine_Tool_Inspection_and_Acceptance_Testing_01.pdf [14.4.2020].

Hummel, V., Ranz, F. and Sihn, W., 2017. *Capability-based Task Allocation in Human-robot Collaboration.* doi: 10.1016/j.promfg.2017.04.011.

ISO 8373:2012 *Robots and robotic devices – Vocabulary.* ISO, 2012

ISO 9283:1998 *Manipulating industrial robots. Performance criteria and related test methods.* ISO, Geneva 1998.

Itaya, T., Sugita, S. and Takeuchi, Y., 2004. *Development of robot teaching support devices to automate deburring and finishing works in casting.* doi: 10.1007/s00170-003-1602-5.

Larkin, N., Norrish, J., Pan, Z., Polden, J. and Van Duin, S., 2012. *Recent progress on programming methods for industrial robots.* doi: 10.1016/j.rcim.2011.08.004.

Li, Y. and Xu, Q., 2009. *Dynamic modeling and robust control of a 3-PRC translational parallel kinematic machine.* doi: 10.1016/j.rcim.2008.05.006.

Li, Y. and Xu, Q., 2008. *Stiffness analysis for a 3-PUU parallel kinematic machine.* doi: 10.1016/j.mechmachtheory.2007.02.002.

Li, Y. and Xu, Q., 2007. *Kinematic analysis of a 3-PRS parallel manipulator.* doi: 10.1016/j.rcim.2006.04.007.

Meyer, C. and Schraft, R.D. *The need for an intuitive teaching method for small and medium enterprises*., 2006. doi: 10.1016/B978-008045157-2/50099-7.

Mutlu, L. and Uyar, E., 2012. *Modelling and kinematic analysis of a built up linear delta robot.* doi: 10.4028/www.scientific.net/AMM.186.234.

Oriolo, G., Sciavicco, L., Siciliano, B. and Villaniand, L., 2009. *Robotics Modelling, Planning and Control.* Springer, (644) p. ISBN 978-1-84628-641-4

Park, F.C. and Lynch, K.M., 2017. *Modern robotics: mechanics, planning, and control.* Cambridge University Press, (624) p. ISBN 1107156300.

Piszczek, L., Płaczek, M., 2018. *Testing of an industrial robot's accuracy and repeatability in off and online environment.* doi: 10.17531/ein.2018.3.15.

Siciliano and Khatib., 2008. *Handbook of robotics.* Springer, (1628) p. ISBN: 978-3-540-23957-4

Stamper, R.E., 1997. *Thesis Report Ph.D., A three degree of freedom parallel manipulator with only translational degrees of freedom.* Available at:

https://drum.lib.umd.edu/bitstream/handle/1903/5902/PhD_97-4.pdf?sequence=1 [10.4.2020].

Zheng, Z., Zhang, X., Chang, J.Z.Z., 2015. *A stable platform to compensate motion of ship based on stewart mechanism.* doi: 10.1007/978-3-319-22879-2_15.

Appendix 1. Solving the unknown joint variables from the inverse kinematics equations.

Variable $\gamma_1$can be solved from the equation (23),

$$\gamma_1 = \text{acos}(\frac{-\overline{A_1C_1}\cdot \text{s}(\theta) - \boldsymbol{Z}}{\overline{C_1B_1}}) \tag{34}$$

Variable $\alpha_1$can be solved from equation (22),

$$\alpha_1 = \text{asin}(\frac{-\overline{OA_1}\cdot \text{s}(30) + \overline{A_1C_1}\cdot \text{c}(\theta)\cdot \text{s}(30) + \overline{B_1P}\cdot \text{s}(30) - \mathbf{Y}}{-\overline{C_1B_1}\cdot \text{s}(\gamma_1)}) - 30 \tag{35}$$

Equation (21) can be presented as,

$$\overline{A_1C_1} = \frac{\overline{OA_1}\cdot \text{c}(30) - \overline{C_1B_1}\cdot \text{s}(\gamma_1)\cdot \text{c}(30+\alpha_1) - \overline{B_1P}\cdot \text{c}(30) - \mathbf{X}}{\text{c}(\theta)\cdot \text{c}(30)} \tag{36}$$

Now the equations (34) and (35) can be placed in the equation (36), which describes the inverse kinematics solution to the first screw axis as a function of x, y and z coordinates. This form is efficient when creating the control logic of the delta robot with the PLC. The inverse kinematics equations to the second and third screw axis can also be solved similarly.

Variable γ_2 can be solved from the equation (26),

$$\gamma_2 = \text{acos}(\frac{-\overline{A_2C_2}\cdot \text{s}(\theta) - \boldsymbol{Z}}{\overline{C_2B_2}}) \tag{37}$$

Variable α_2 can be solved from equation (24),

$$\alpha_2 = \text{acos}(\frac{-\boldsymbol{X}}{\overline{C_2B_2}\cdot \text{s}(\gamma_2)}) \tag{38}$$

Equation (25) can be presented as,

$$\overline{A_2C_2} = \frac{-\overline{OA_2} + \overline{C_2B_2}\cdot \text{s}(\gamma_2)\cdot \text{s}(\alpha_2) + \overline{B_2P} + \mathbf{Y}}{-\text{c}(\theta)} \tag{39}$$

Placing the equations (37) and (38) to equation (39) yields the inverse kinematics solution to the second screw axis as a function of x, y, and z position.

Variable γ_3 can be solved from the equation (29),

$$\gamma_3 = \mathrm{acos}\left(\frac{-\overline{A_3C_3}\cdot \mathrm{s}(\theta) - \mathbf{Z}}{\overline{C_3B_3}}\right) \tag{40}$$

Variable α_3 can be solved from equation (28),

$$\alpha_3 = \mathrm{asin}\left(\frac{-\overline{OA_3}\cdot \mathrm{s}(30) + \overline{A_3C_3}\cdot \mathrm{c}(\theta)\cdot \mathrm{s}(30) + \overline{B_3P}\cdot \mathrm{s}(30) - \mathbf{Y}}{-\overline{C_3B_3}\cdot \mathrm{s}(\gamma_3)}\right) - 30 \tag{41}$$

Equation (27) can be presented as,

$$\overline{A_3C_3} = \frac{\overline{OA_3}\cdot \mathrm{c}(30) - \overline{C_3B_3}\cdot \mathrm{s}(\gamma_3)\cdot \mathrm{c}(30+\alpha_3) - \overline{B_3P}\cdot \mathrm{c}(30) + \mathbf{X}}{\mathrm{c}(\theta)\cdot \mathrm{c}(30)} \tag{42}$$

Placing the equations (40) and (41) to the equation (42) gives the inverse kinematics solution to the third screw axis as a function of x, y and z coordinate point.

Appendix 2. The MATLAB script for inverse kinematics equations.

```
syms A1C1 A2C2 A3C3 x y z
OA1      = 281;      %length in mm
OA2      = OA1;
OA3      = OA1;
C1B1     = 532;      %length in mm
C2B2     = C1B1;
C3B3     = C1B1;
B1P      = 50;       %length in mm
B2P      = B1P;
B3P      = B1P;
theta1   = 47;       %angle in degrees
theta2   = theta1;
theta3   = theta1;
z_offset = 541.3;    %Origin O(x,y,z) is moved to -z-direction by 541.3mm


eqn1 = (OA1*cosd(30)-C1B1*sind(acosd((z_offset-A1C1*sind(theta1)-
z)/C1B1))*cosd(30+(asind((-
OA1*sind(30)+A1C1*cosd(theta1)*sind(30)+B1P*sind(30)-y)/(-
C1B1*sind(acosd((z_offset-A1C1*sind(theta1)-z)/C1B1))))-30))-
B1P*cosd(30)-x)/(cosd(theta1)*cosd(30)) == A1C1;

eqn2 = (-OA2+C2B2*sind(acosd((z_offset-A2C2*sind(theta2)-
z)/C2B2))*sind(acosd((-x)/(C2B2*sind(acosd((z_offset-A2C2*sind(theta2)-
z)/C2B2)))))+B2P+y)/(-cosd(theta2)) == A2C2;

eqn3 = (OA3*cosd(30)-C3B3*sind(acosd((z_offset-A3C3*sind(theta3)-
z)/C3B3))*cosd(30+(asind((-
OA3*sind(30)+A3C3*cosd(theta3)*sind(30)+B3P*sind(30)-y)/(-
C3B3*sind(acosd((z_offset-A3C3*sind(theta3)-z)/C3B3))))-30))-
B3P*cosd(30)+x)/(cosd(theta3)*cosd(30)) == A3C3;


solA1C1 = vpa(solve(eqn1,A1C1),10)
solA2C2 = vpa(solve(eqn2,A2C2),10)
solA3C3 = vpa(solve(eqn3,A3C3),10)
```