Tampereen ammattikorkeakoulu

Simulation of a Robot Arm in Reconfigurable Workspace

Jarno Heikkilä

OPINNÄYTETYÖ
Joulukuu 2022

Konetekniikka
Älykkäät koneet

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Konetekniikka
Älykkäät koneet

HEIKKILÄ, JARNO:
Robotin simulointi uudelleen konfiguroitavassa työympäristössä

Opinnäytetyö 46 sivua, joista liitteitä 0 sivua
Joulukuu 2022

Työn tavoitteena oli selvittää lukijalle, mitä tarkoitetaan konfiguroitavalla työympäristöllä ja mihin aspekteihin olisi syytä kiinnittää huomiota. Konfiguroitavassa työympäristössä tarkoituksena oli tarkastella sen muokattavuutta.

Työ tehtiin Cargotec Oyj:n toimeksiannosta. Opinnäytetyön kirjoituskielenä oli englanti. Simulaation tekemiseen käytettiin RoboDK-sovellusta. Kyseinen sovellus valittiin työn ohjaajan suosituksen perusteella. Perusteluna ohjelman valinnalle oli lisensseihin liittyvä saatavuus.

Työn alussa esitellään terminologia. Työssä aluksi esitellään simulaatio ja siihen liittyviä ongelmia. Normaalista robottisolusta poiketen työssä itse robottia liikutetaan ennen kuin se aloittaa normaalin työrutiininsa. Tämä tarkoittaa työympäristön uudelleen konfigurointia.

Teoriassa esitellään muutamia kaavoja robottien kinematiikkaan liittyen. Suorassa kinematiikassa lasketaan robotin viimeisen nivelen paikka ja orientaatio. Käänteisessä kinematiikassa lasketaan nivelten kulmat. Muita tärkeitä kinematiikan aiheita robotiikassa ovat Jacobian- ja Denavit-Hartenberg-matriisit. Turvallisuusstandardeista esitellään robotteihin liittyvät pääpiirteet.

Teorian jälkeen esitellään, kuinka RoboDK-sovellusta yleisesti ottaen käytetään. On tärkeää tietää, kuinka simulaatioon voi tuoda erilaisia objekteja, joko itse tehtyjä tai sovelluksen kirjastosta löydettyjä. Robotin käyttöön liittyen pitää tietää, kuinka luodaan liikeratoja robotin toiminnalle.

Työn ratkaisussa kerrotaan mahdollisimman tarkkaan, kuinka simulaatiotilanne on ratkaistu. Halutussa simulaatiossa pitää robotti ja sen työpiste tuoda ensin lähemmäksi toisiaan, jonka jälkeen robotti tekee työrutiininsa. Työkalujen liittäminen robottiin ja niiden irrottaminen ei ollut yksinkertaista.

Asiasanat: robodk, simulaatio, robotiikka, muuttuva työympäristö

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Mechanical Engineering
Intelligent Machines

JARNO HEIKKILÄ:
Simulation of a Robot Arm in Reconfigurable Workspace

Bachelor's thesis 46 pages, appendices 0 pages
December 2022

This thesis was commissioned by Cargotec Corp. The objective for this thesis is to show to the reader how to reconfigure workspace for the robot in a simulation. Simulation itself will not show or comment any real-world problems. Simulation was created with RoboDK software. Thesis advisor suggested that software because TAMK has a license for it.

Simulation situation and problems are defined first. Then, in theory, forward and inverse kinematics in robotics are presented by showing the main formulas for those theories. Some interesting issues concerning safety standards related to robotics were listed.

RoboDK software basic functions are introduced to the reader before the solution for the simulation is gone through the complete solution. Solution is told as accurately as possible. The robot manipulator reconfigured its workspace before the robot did its job routines. This reconfiguration was made with linearly moving platforms that were found in the RoboDK library. Hardest part in the simulation was to avoid singularity situations. Singularity is a situation where the manipulator will have unwanted sudden movement of one of its joints.

Key words: robodk, simulation, reconfigurable workspace, robotics

TABLE OF CONTENTS

1 INTRODUCTION 7
2 PRESENTING THE SITUATION 8
2.1 Simulation 8
2.1.1 Benefits of simulation 8
2.1.2 Purpose for the simulation 9
2.1.3 Problem definition 10
2.1.4 Reconfigurable workspace 11
2.1.5 Simulation scenario 11
2.2 Software 13
3 THEORY 14
3.1 Forward Kinematics 14
3.1.1 Denavit-Hartenberg matrix 14
3.2 Jacobians 15
3.3 Inverse Kinematics 16
3.4 Manipulator selection factors 16
3.4.1 Degrees of freedom 17
3.4.2 Reachability 17
3.4.3 Workspace design 19
3.4.4 Safety standards 19
4 SIMULATION 21
4.1 RoboDK UI 21
4.2 Manipulator selection 22
4.3 Reference Frames and Target Points 23
4.3.1 Programs 24
4.3.2 Custom Tool 25
4.4 Movements and singularities 25
4.5 Environment 26
4.5.1 Working with Tool Changers in RoboDK 26
4.5.2 Creating environment 27
4.5.3 Simulation 29
4.5.4 Reconfiguring workspace 29
4.6 Exporting results 30
5 SOLUTION 31
5.1 Designing simulation in RoboDK 31
5.1.1 Simulation plan 32
5.1.2 Moving workspace in the simulation 35

5.1.3 Tool starting positions 35
5.1.4 Checking reachability 37
5.1.5 Singularities 40
5.1.6 Designing movements 41
5.1.7 Attach and detach tools 44
6 DELIBERATION 45
SOURCES 46

ABBREVIATIONS AND TERMS

OLP	Off-line programming
TCP	Tool Centre Point
DH	Denavit-Hartenberg
RoboDK	Robot Development Kit
DOF	degree of freedom
SCARA	Selective Compliance Assembly Robot Arm
DES	Discrete-event simulation
OR	Operational research
Corp.	Corporation
End-effector	at the free end of the chain of links that make up the manipulator
Elbow	Joint 2 orientation relation
Front	Joint 1 orientation relation
Non-flip	Joint 5 orientation relation
Master	Tool changer attached to robot
Slave	Tool changer attached to tool
Reference frame	A local coordinate system for selected object related to world origin coordinate system
Wrist frame	Affixed to the last link of the manipulator
Payload	the weight the robot can lift
Singularity	a configuration in which the robot end-effector becomes blocked in certain directions
Target	Recorded position for the robot
Parent-child	A tool is a child for the robot
Tool	An object that is attached to the robot
Tool Centre Point	Reference frame in the tool
Workspace	Envelope where a robot manipulator can operate
Work area	Place where tools will be placed

1 INTRODUCTION

Simulating a robot manipulator in a reconfigurable workspace is interesting, because usually a robot has guaranteed workspace that doesn't change. In this thesis, the simulation will consist of a simple configuration before the manipulator can do their job. The manipulator and the work area will get closer with the help of two linearly moving mechanisms. This configuration is done because without it the robot does not have enough reach to make the intended movements.

Knowing the basics of kinematics in robotics will be helpful to understand what is happening in the simulation. Kinematic formulas will help the reader to understand how you would calculate joint positions and angles in mathematical form. When using simulation software, you will see that knowing kinematics will help when designing and constructing movements. Safety standards are there to make sure that robotics designers will remember to use safety walls as necessary. In this simulation scenario it is not necessary to create these walls, because there should not be any people walking around when the robot is taking its actions.

The main objective for this thesis is to show how you would simulate a robot manipulator with the selected software. Selected simulation software is RoboDK. This simulation can be used as a basis scenario for the actual simulation that my workplace wants. For this reason, this simulation will also be a very simplified design. As this simulation can or will be the basis of actual simulation of a real-world situation, I cannot use the actual environmental objects. I have to design much simplified versions of those objects or just leave those spots blank.

This thesis will follow the following pattern, first answer questions about this subject. Then the basic theories will be presented. Question of how will be answered after the theory section. Basic functions of the simulation software is presented to the reader before the solution for my simulation situation is written as accurately as possible. Deliberations come after the solution and have some thoughts about the simulation and the software itself.

2 PRESENTING THE SITUATION

What is simulation? Why this subject as a thesis? What is the situation that is to be simulated and what problems there could occur? Those are some of the many questions that this chapter is trying to answer. Answering those questions should introduce the subject of this thesis properly.

2.1 Simulation

“Simulation is an imitation of the operation of a real-world process over time.” “Simulation is an indispensable problem-solving methodology for the solution of many real-world problems.” (Banks J., 1998, 3) This means that you can aid your design work with simulations to see possible problem situations. “Discrete-event simulation, DES, is seen as the mainstream simulation approach in the field of operational research (OR).” OR specialists ignore the system dynamics simulation as a path in simulation, for them DES is what a simulation is. (Brailsford S., Churilov L., Dangerfield B., 2014, 10).

In this thesis, a specific scenario will be simulated. Simulation will be done with my personal computer that has simulation software capable of simulating robots. Simulation software will be RoboDK. RoboDK is an abbreviation from Robot Development Kit.

In my simulation case the robot and the work area are at first away from each other, but then they will get closer and then the manipulator will do its job routine and finally the manipulator and the work area will separate from each other again. As I present the solution for the simulation, I will bring all the aspects that were used in RoboDK.

2.1.1 Benefits of simulation

Simulation software, RoboDK, has almost 700 different 6 degrees of freedom robot manipulators in its library. In the library, there are over 100 different ready-made tools. Using products from the software’s own library is beneficial because in these products software designers have already made the mechanical

constructions that represent real world products. Dimensions of the robot manipulators should therefore be correct.

Simulating any situation can be beneficial when deciding what products are good for a company's needs. Let's say for example that an engineer has a task to design a manufacturing cell using a robot manipulator. In this case, the engineer can compare different manipulator models before presenting the best solution for his or her bosses. This can be called financial and a time benefit. Financial benefit with the aspect that after simulating the situation the selected robot will be able to handle the situation that it was bought for. Time benefit is apparent in situations where simulations are as close as possible in comparison to the real-world situation. RoboDK is offline programming software, OLP, so it can translate the simulated situation for the actual robot in robot language.

Other benefits can be locating possible bottlenecks in manufacturing, exploring what if situations and keeping engineers' knowledge up for current models of robots. Simulation in RoboDK shows robot arm trajectories when it makes its movements. Making sure that these trajectories are inside a certain area can be thought of as a safety check. This will of course be checked in the actual real-world situation.

2.1.2 Purpose for the simulation

Simulation is a tool for showing how movements would happen in a real-world situation. This simulation will be a simplified version of the future product Cargotec Corp. will possibly have. Simplifications are because, I should not show an exact situation of the real-life situation and, because the product that this simulation is for is not yet ready and most of the aspects of it are secrets including the adapters that I designed earlier this year.

After this thesis work, I will implement this simulation scenario for Cargotec Corp. in the future product design simulation. This simulation that I will make for Cargotec Corp. will be made with the same RoboDK software.

I have simulated fluid dynamics and some mechanical machines in SolidWorks, but to simulate a whole work environment and robot manipulator is new to me. So, as I write this thesis, my hope is that I will learn plenty of process simulation, the simulation programming and different aspects of including the environment as part of the simulation.

2.1.3 Problem definition

Simulation that is made for the thesis can be thought of as a two-fold problem. Firstly, how to bring a robot and work area closer together and secondly how to install tools in the right places with the help of tool changers. Moving the robot and work areas closer together is also called reconfiguring the workspace. Before I started using RoboDK, I read some posts from the internet about the tool changer operations. In 2018, they said that they were writing macros to make the tool changer operations smoother. Suggestions that time was to copy tools and place them at the start and final locations, then hide the tool based on robot movements. (Automatic tool changers in RoboDK, RoboDK-forumpage)

Simulation will be planned in a way that there will not be any singularities in the manipulator. There will be 3 tools to install into a so-called work area wall, two of these tools will have a 90-degree rotation around the z-axis at the end. The robot will make its movements in Cartesian space. The robot and the work area will have their own moving mechanisms. Movement in these mechanisms is linear.

I shall use a specific robot manipulator in the simulation, that is not necessarily the same as Cargotec Corp. would use in their product. The manipulator I will select with its reach and weight in mind. Moving mechanism platforms will be whatever can be found in the software library. Also, tool changers will be selected from the library, and they will be in the same series, so no two different manufacturers for tool changers, but with the aspect which one looks the best on the manipulator. I found out from the library that tool changers and manipulators do not have any compatibility information in them. That is something that I shall investigate from tool changer and manipulator manufacturers' websites.

The manipulator wrist flange has a certain dimension for holes in the end-effector. The tool changer has to have same dimension for its installation holes.

2.1.4 Reconfigurable workspace

Reconfiguration of the workspace means in this simulation case that the work area and the robot will get closer to each other before the robot can start its actual job tasks. As the robot completes its tasks then the work area will be moved back. This means that tools are left in the work area and then the robot workspace and the work area move away from each other.

Some aspects come to my mind when thinking about problems that could occur when the workspace is not static when using robots. One is that, are the movements to configure the workspace always accurate? Errors in the configuration accuracy will cause problems for the robot manipulator unless it is using a vision system to compensate for small errors. The wear and tear can also cause these issues in workspace configuration accuracy. In this simulation, the manipulator will always be in a static place so these accuracy issues will not come apparent in the simulation.

The robot picks up objects and installs them in the work area that has come closer at the beginning of the simulation. Objects will have predetermined locations in the work area for every tool. In the simulation it is important to avoid collisions. To do this I have to design some movements differently than others. I will show in the simulation how to rotate an object 90 degrees sideways. This movement cannot be a joint movement, it has to be linear. This means that the object translates linearly.

2.1.5 Simulation scenario

The designed simulation for this thesis has many steps, some of them are similar in comparison with the previous steps. I have mentioned some simulation steps already. First is the reconfiguration of the workspace. World origin will be in the middle of this work area, visually the world origin will be on the floor. The

manipulator will make its movements in the coordinate system that world origin is in, but the movements will be approximately 1.5 m above ground.

After the environment changes, the robot can start to make its movements. Movements are programmed in a specific order. This way you can escape the singularity situations more easily. Basic movement will be listed here. In the solution, this list will be more accurate. Placing tools must be done in order of 1 to 3. If simulation would continue and the tools would also be put back in the tool stand then this order can be done with the same 1 to 3 instruction or 3 to 1. In the simulation, the order is not important, but in the real-world situation tools would have pneumatic tubing attached to them and entanglement might become an issue if order is mixed between attaching and detaching actions.

Singularities might be a problem when the arm is approaching placement location. This is something to keep an eye on. Finding a solution to the final movement depends on the factor if the tools can have a parent-child connection with the work area.

1. Bring the work area and robot closer with each other
2. The robot arm picks up the first tool
3. Place it in the work area and leave it there
4. Pick second tool
5. Place it and leave it
6. Pick third tool
7. Place it in work area and leave it
8. Move the robot to a so-called home position
9. Environment changes back to original conditions.

Simulation finishes when the environment has changed back to its original state. Wrist singularities might be an issue in two places, when picking up a new tool and when leaving this tool in. If singularities occur, then I will design the workspace so that these problems will vanish. After rotating tools in place there will also be a moment where there is a risk of singularity, this singularity would occur in the so-called elbow joint.

2.2 Software

I have not had a chance to work with robot simulation software before. My Line Manager firstly suggested that Visual Components would be the software that this simulation would be done with. Licensing issues with that software meant that this software cannot be a valid choice. TAMK has a licence for RoboDK software. My thesis advisor suggested that this would be a good and valid choice for the simulation software. I made the simulation with the 30-day trial version of the software. The trial version has a limitation of 50 lines of code in the simulation, also you cannot access the drivers for the robots, but you will not need them in the simulation. Exporting complex solution is not possible with the trial version of the software. (RoboDK-pricing webpage)

RoboDK comes from the expression Robot Development Kit. Software company is based in Montreal, Canada. RoboDK is Off-Line Programming (OLP) software. It also has capabilities to work as Online Programming software. OLP allows you to create from simulation in RoboDK a program that the manipulator can understand. No connection between software and the robot is needed for OLP. On the other hand, if you use RoboDK as online programming software, then you must connect the software and the manipulator. In this thesis, the focus is on the simulation part of the software. It would not be hard to transfer this simulation to a robot.

Other features that this software has includes an extensive library where there are robots from more than 40 different companies, the robot machining and exporting your programs to robots. You can use your robot like a 5-axis CNC machine with simulating and converting NC programs to your robot in RoboDK. Exporting programs to your robot will save time, when you do not have to teach target points again, as you will use the target points from the simulation. (RoboDK main webpage)

3 THEORY

In this chapter, I will summarize a few theory subjects that are interesting in robotics, these include Forward Kinematics, Jacobians, and Inverse Kinematics. I have used the same Introduction to Robotics book by J.J. Craig as the main source material. Safety standards are gone through quickly and mainly based on two different standards. I will also go-through some aspects that I thought about when I selected the robot and other objects for the simulation.

3.1 Forward Kinematics

“Kinematics is the science of motion that treats the subject without regard to the forces that causes it”. A position, velocity, and acceleration are the studied objects within kinematics. In forward kinematics you calculate the position and orientation of the end-effector. (Craig, J.J., 2005, 4-5, 62)

In kinematics connecting links in the robot has only two quantities that need worrying: a link offset and a joint angle. Link offset is distance along the common axis from one link to a next link. The joint angle is how much it has rotation along this same common axis. You need the link length and the link twist parameters to fully describe situation in kinematic form. (Craig, J.J., 2005, 64-66)

Basic form for the transformation matrix is following:

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

where c is cosine, s is sine, θ is joint angle, α is link twist and d is link offset distance.

3.1.1 Denavit-Hartenberg matrix

The Denavit-Hartenberg matrix is defined by the means of the link parameters. The notation of this matrix gives a standard methodology to write the kinematic equations of a manipulator. In manipulators, this is practical because with this

matrix you represent the pose of one body with respect to another. (Craig, 2005, 67, 69) The matrix has a following representation:

$$M_{n-1,n} = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} & T_x \\ R_{yx} & R_{yy} & R_{yz} & T_y \\ R_{zx} & R_{zy} & R_{zz} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos\theta_n & -sin\theta_n & 0 & a_{n-1} \\ sin\theta_n cos\alpha_{n-1} & cos\theta_n sin\alpha_{n-1} & -sin\alpha_{n-1} & -d_n sin\alpha_{n-1} \\ sin\theta_n sin\alpha_{n-1} & cos\theta_n sin\alpha_{n-1} & cos\alpha_{n-1} & d_n cos\alpha_{n-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Link parameters are used in the convention of DH notation. In one joint there are 3 parameters, so in a 6 DOF robot manipulator will have 18 different parameter values to define the DH-matrix. (CRAIG, 2005, SIVUNROT)

3.2 Jacobians

In the motion analysis of the manipulator, you need to examine the linear and angular velocity of a rigid body. Considering the forces acting on the rigid body are used in the application of static forces with manipulators. The matrix that uses these velocities and static forces is called Jacobian. The specification of this Jacobian is that it is the Jacobian of the manipulator. Jacobian is a multidimensional form of the derivative. (Craig, 2005, 135, 149)

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Jacobian matrix can be written as above. Functions inside the matrix are partial derivatives. The matrix usually in the robotics is even a number matrix. So, if a robotic manipulator has 6 degrees of freedom, then Jacobian matrix is 6×6.

Jacobian matrix can be used to calculate singularities. The matrix cannot be inverted at singular point. To find this singular point examining the determinant of Jacobian is a must. The determinant is zero, where singularity exists. There are two types of singularities: workspace-boundary and workspace-interior.

Boundary singularity occurs when the manipulator is fully stretched out in a way that the end-effector is near the boundary. Also, being folded back on itself is boundary singularity. Lining up two or more joint axes will cause workspace-interior singularity. (Craig, 2005, 151-152)

3.3 Inverse Kinematics

In forward kinematics computing the position and orientation of the end-effector is the main issue. In inverse kinematics, you calculate from those positions and orientations all possible sets of joint angles. Wrist frame must be found with frame transformations. The wrist frame is relative to the base frame of the robot. (Craig, 2005, 101)

Inverse kinematics is non-linear mathematics problem. This means that the equations might be hard to solve. (Craig, 2005, 102) Industrial robots have goal points that can be taught to them. Teaching goal points are done by physically moving the manipulator. By teaching goal points to the robot and then playing them back when needed, does not involve inverse kinematics. This is because goal points were not defined in the Cartesian space. (Craig, 2005, 127)

You will have at least two ways of solving inverse kinematics equations: algebraic or geometric. In geometric solution, we select a plane, for example xy-plane, and solve the joint angles using geometric mathematics. Algebraic solution is not as simple because in some equations you must check that you could even calculate it. In algebraic mathematics, there are transcendental equations. These can be simplified using polynomials. If the solution is complex, then there is no real solution to the original transcendental equation. (Craig, 2005, 110-114)

3.4 Manipulator selection factors

When making a purchase order of a robot manipulator for your company, you should be aware of certain factors. Subjects that follow are the most common selection factors.

Simulation can help when selecting a manipulator for your application, because with it you can check some limitations that manipulators have, and you can also check the compatibility of other objects with the manipulator. Some simulation software are also capable of creating robot language code, this includes RoboDK. This means that you do not have to code your robot again if your simulation is accurate enough.

3.4.1 Degrees of freedom

Degrees of freedom, DOF, determine how nimble the manipulator can be. If the movements are simple, then you will not need as many DOF as in more sophisticated workspace. The work area and motions related to this work area can determine the type of manipulator that is needed. SCARA robots are a valid subject for consideration if the manipulator does its work in a level platform.

Commonly, the robots have 6 DOFs. In RoboDK library there are 5, 6 and 7 DOF and SCARA robots. A SCARA robot is flexible in XY-plane, but rigid in the Z-axis, this is where the SCARA comes, which is Selective Compliance Assembly Robot Arm. The SCARA robots are useful for example in situations where you move objects from one place to another. (SCARA robots, Fanuc website)

3.4.2 Reachability

Ying and Iyengar (1995) say in their article that definition of reachability in robotics is the ability of the robot manipulator to move its joints and links in a free space for the hand of the manipulator to reach the given target.

Robot manufacturers tell in data sheets how far the robot manipulator can reach. In the data sheet, there is also a number for payload in kilograms. The robot can move the weight of this payload amount as far it can reach. On the other hand, the manipulator will lose its accuracy at the end of its reachability eventually because of fatigue. Simulation software shows the reachability of the manipulator differently, it shows a 3D-envelope over the manipulator. Following images are examples of Fanuc CR-15iA robots' workspace and how they are shown in data sheets and in the RoboDK simulation.

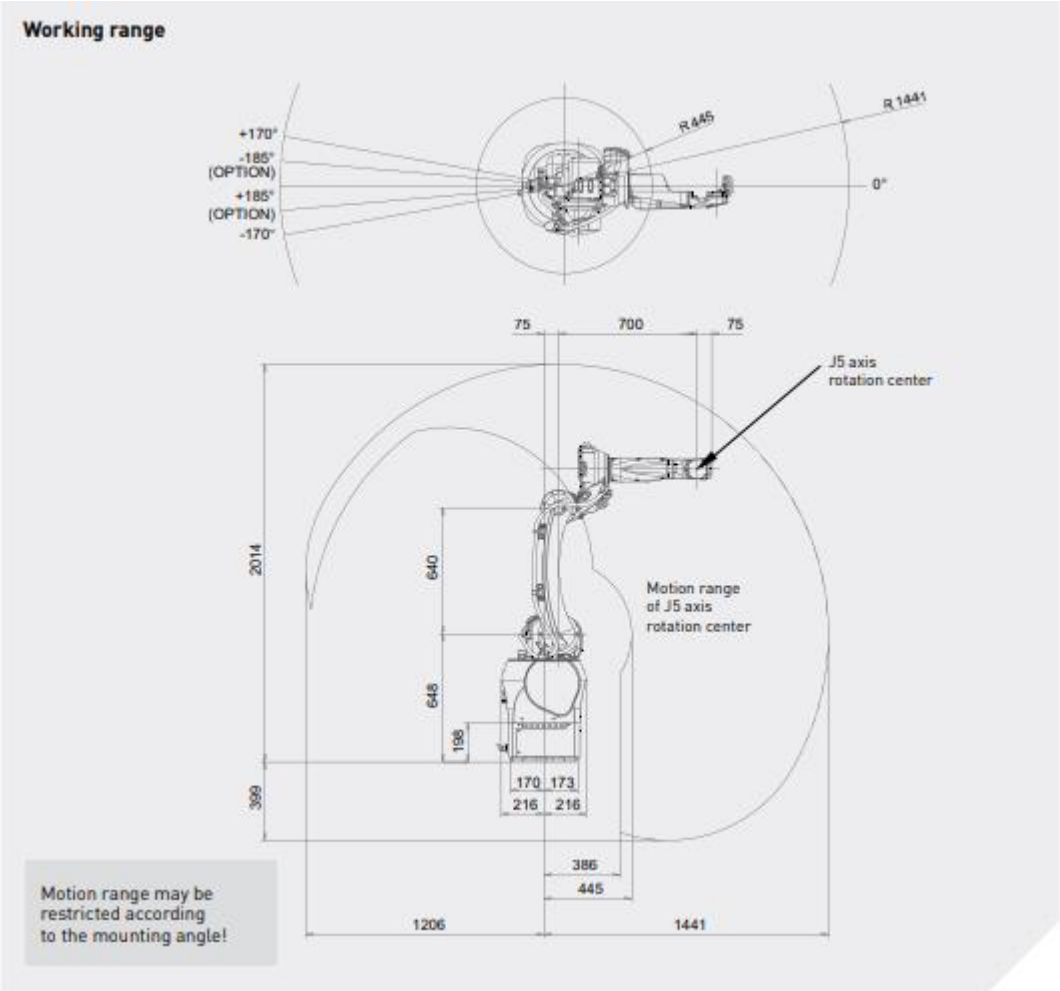

Figure 1. CR-15iA Collaborative robot working range from the data sheet

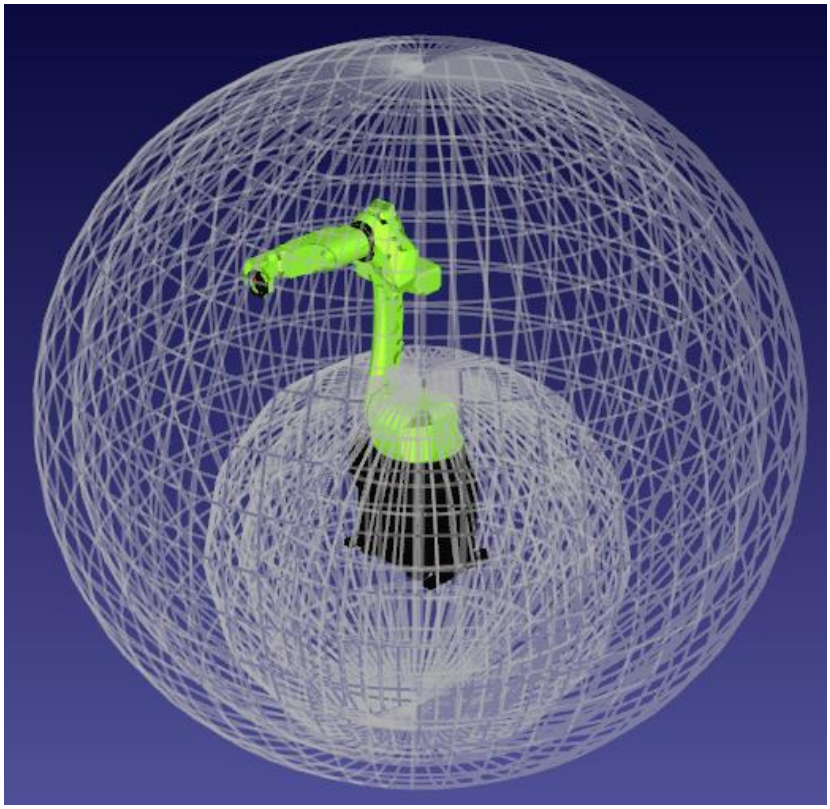

Figure 2. Fanuc CR-15iA workspace for tool configuration from RoboDK

3.4.3 Workspace design

In this simulation, the workspace is reconfigured every time at the start of the simulation. In a simulation environment reconfigured changes are always the same. The robot manipulator has a certain workspace envelope. Careful design needs to be done when designing workspace, because singularities and tight workspace might be an issue.

It is helpful to know the kinematics in robotics before you start the purchase process of your manipulator. This is because that way you are not likely to buy something extra that is not eventually needed. Some design questions can be checked in the simulation software. It is beneficial to check the types of movements the robot needs in its workspace as an example. This way you can check if a robot with fewer DOF can do the same work and if the size of the robot is sufficient.

3.4.4 Safety standards

I shall express the most important issues related to safety standards. There is plenty more safety issues in the standards that you should consider before starting to use a robot.

Emergency stops must always be reset manually, protective stops can be reset automatically. As a general requirement, a robot cannot have any motions when rebooting the power after a power loss event. No unintended operations or movements are allowed. Safety functions must be operational even if a single fault occur in the safety related parts. After this fault, a safety operation will be performed every time and the robot must be in a “safe” mode until the fault is fixed. The robot must have one or more emergency stop functions and one or more protective stop functions. If the robot is used in a reduced speed, then it must not exceed the speed of 250 mm/s in TCP. In a singularity situation, the robot can have unexpected high speeds. The robot has to give some warning to the user if this situation occurs. (ISO 10218-1)

Eliminating possible hazards is important. A risk assessment process is necessary after the hazards are identified. Measures to eliminate the risks are done after the risk assessment process. The robot system is designed in a way that exposing personnel to hazards are avoided. Protective measures for the robot must consider environmental conditions. A single emergency stop function will stop the motion in the robot and in other related functions that can be considered hazardous. Loss of power will not have an effect in the end-effector. Safeguarded space for the robot must be established. This can happen with physical limitations or a dynamical limitation. A light curtain is an example of a dynamic limitation. Using limiting devices will make the space where the robot does its task routine restricted. (ISO 10218-2)

4 SIMULATION

Simulation will be done with RoboDK simulation software. I will simulate how a robot manipulator moves its end-effector from a target point to a target point in the environment, similar point-to-point motion is meant to be done with the workspace as well.

This simulation will be discrete-event simulation (DES). DES is used to model systems that can be viewed as queuing network. An example of a queue in my simulation is that I must move tools one by one, and I cannot move them all at the same time. (Brailsford, Churilov, Dangerfield, 2014, 12-13)

4.1 RoboDK UI

Figure 3. RoboDK UI

Buttons in Figure 3 are following from left to right (if a button has a shortcut, it is expressed in parenthesizes):

1. Open
2. Library
3. Save
4. Undo
5. Redo
6. Reference Frame
7. Target
8. Fit All
9. View Angle (Isometric as default)
10. Don’t move reference frames, objects, or tools
11. Move reference frames keeping the relative position with the child items (hold Alt)
12. Move a robot tool (TCP) with respect to the robot flange. (hold Alt+Shift)
13. Check Collisions
14. Fast simulation (space key)
15. Pause the program (Backspace key)

16. Add Python Program
17. A new program linked to the selected robot
18. Add joint movement
19. Add a linear movement
20. Add circular movement
21. Add a pause to program
22. Show a message
23. Program call or insert message
24. Sets specific output
25. Add an event to be executed offline only
26. Export

It is important to know where you can find the following items in the UI: Reference Frame, Target, and Programming. Also, the library for robot manipulators is good to know where it is.

4.2 Manipulator selection

Workspace for the robot is 1.8 m in length and 2.3 m in width. This is the area where the robot can be placed. Small robots normally have a reach around 1.4 m. Placing the robot in a right place is important.

Accuracy in movements is important. More degrees of freedom, DOF, means that the manipulator is more able to move around obstacles. A robot manipulator with 6 DOF is the most common manipulator.

I have selected for the simulation KUKA KR 10 R1420 robot manipulator. This model had the best reachability to robot weight ratio. Weight of the robot is around 150 kg. Other models had approximately 100 kg more weight. Payload with all the robots is similar, 10 kg. Payload is the weight the robot can handle in its fully extended condition. Pose repeatability is based on the data sheet ±0,04 mm.

This manipulator also has a following mounting flange:

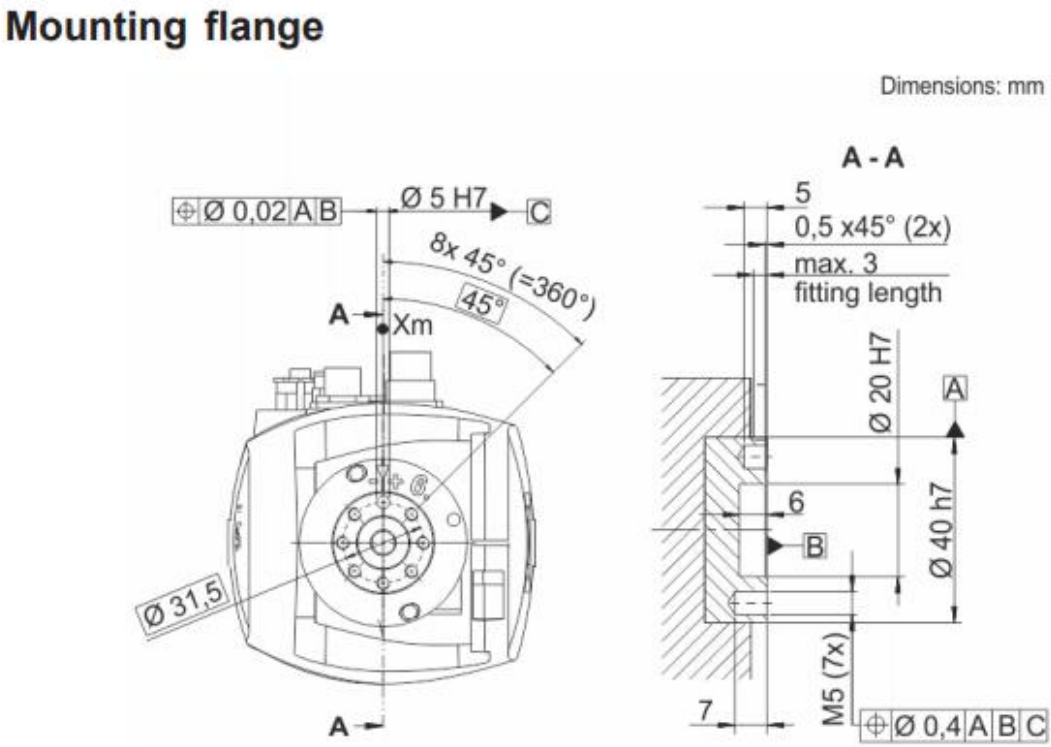

Figure 4. KUKA KR10 R1420 Mounting flange dimensions

Based on the mounting flange figure you should select a tool changer that has 31.5 mm diameter for the screw holes. RoboDK library does not inform the user for those diameter dimensions. User must check these dimensions manually from the manipulator manufacturers' webpage. Tool changer dimensions must be checked manually as well.

4.3 Reference Frames and Target Points

Reference frames are always related to the world origin in a Cartesian space. The robot base has a reference frame, work area and tool stand also should have reference frames. Tools have a Tool Center Point, TCP. It is in the tip of the tool. Reference frames are moved into the robot's workspace to allow the robot to do its task programme.

Target points are locations the robot's end-effector moves between. In the simulation target points must be taught one by one. To teach a target point, you must select the related target options for the movement and move the end-

effector to a desired location. While you do this checking that the robot has consistent configurations between movements is important for simulation consistency. Target points are related to the world origin in the Cartesian coordinate system. The robot can make its movements inside its workspace therefore also the target points have to be inside the workspace. Some movements will still be impossible in the workspace, singularities are usually a cause for these stops in the simulation. Simulation software will not move the end-effector between target points if there is singularity in the motion. Naming target points helps the user to keep them in order.

Reference frames and targets are colour-coded in RoboDK. This means that every axis has predetermined colour. Keeping axes in similar orientation will help with the simulation consistency.

TCP is in the tip of the tool at the centre of the tool symmetrically. One of the axes should point away from the tool. RoboDK will give some distance for the TCP, but this is likely wrong. TCP must be moved related to the tool changer master part. This way the tool changer knows the TCP when moving the tools in the workspace.

4.3.1 Programs

Programs are used to move the robot between two target points. Attach and detach functions are also programs, but they are called events in RoboDK. Selecting a new program from the RoboDK UI will give you a program where there is a new target location, a related tool frame and the type of movement. Knowing that adding a new program gives you target locations is helpful, because you do not want to teach additional target points that are not needed. A wrong tool frame in the program will not allow the program to run.

In an event program, you can call the tool changer to attach to other objects preferably to a tool changer slave part. First you select the function in the event program. In an attach-event you select which tool is attached to the tool changer master part. I had to change the connection distance in the event because

otherwise attachment would not happen. In a detach-event you do not need to select anything.

These smaller movement and event programs will be called in the main program for the simulation. In the main program, smaller programs are called one by one, if you want to move to objects at the same time then you must put two movement programs inside the smaller program. The robot has to be the same if two movements happen at the same time.

4.3.2 Custom Tool

To create a custom tool, you need to import CAD-model from your chosen CAD software. I will use SolidWorks 2020. Importing CAD-models is a quite simple thing as an operation. To do this use the open icon from the UI.

You need to save your CAD-file in right format and then select it from RoboDK. Right formats for any objects that are imported to RoboDK are STEP and STL.

In this simulation, I need to create tools, a tool stand and a work area. The rest of the objects are found in the RoboDK library. I will save files in STEP-format because they are easier to modify compared with the STL-format.

4.4 Movements and singularities

Movements are simulations that are between two target points in a movement program. Keeping an eye on the joint angles is helpful when teaching new target points. You can see these joint angles in the robot control panel in the simulation software. In singularity, one of the joint angles is too close to limit. You can also see if there is singularity as the robot arm changes its configuration suddenly visually.

The simulation software would still inform the user if you missed the sudden movement in the robot arm while teaching a new target point. An error message will be shown while running a program between two target locations. To avoid

singularities, it is helpful for the robot if more target points are taught, and movements made between these points.

You can manually move the robot arm in RoboDK by pressing and holding Alt-button down and grabbing a plane or arrow to move the end-effector. This is also how you can change the position of all reference frames and target points if you want to manually teach new locations. I found that easiest way to teach new target points were in the options of the target. You would use your mouse wheel to scroll a new location. Cursor of the mouse has to be in the coordinate that you want to change. Programs and at the same time the target points should be tested before you run the whole main program simulation.

4.5 Environment

The same way you can bring custom tools to the simulation you can also bring other objects to visually enhance the simulation. In my simulation, I need a tool stand for the tools, work area where to install the tools and objects where the environmental reconfiguration will happen. Environmental reconfiguration means the robot manipulator and the work area have to get closer. Work area must be inside the robot's workspace.

Operation of moving objects on top of custom platforms is not an easy operation. Preferred situation is to find the platforms from the simulation software library. Pre-made platforms will already have designed operational mechanisms. Using these mechanisms is simple. You program a linear movement program. Selecting the right tool frame and robot for this is crucial. To select the robot for a specific program you must right click the program in the tree menu and then change the robot when needed. The right choice in the linear movement of the platform is to select the platform as the robot and not the actual manipulator.

4.5.1 Working with Tool Changers in RoboDK

Tool changers consist of two pieces of equipment: the tool changer master part and the tool changer slave part. One of these is permanently attached to the

end-effector of the robot manipulator and the other is permanently attached to the tool. Every tool needs its own slave part.

To make connection between these two parts you need to program an event where you call an attach-operation between the master and desired slave parts. After moving the tool to the right place, you must remember to call a detach-event program.

4.5.2 Creating environment

The same way that I created the tools in SolidWorks I will also create the tool stand and the work area in that CAD software. I have decided to have a rotation when placing the tools in the work area. This will have an impact in the work-space. Work area has to be inside the workspace for the wrist centre if some tools are rotated to place.

In this simulation tool stand will have three places for three different tools. Tool changer slave parts should be oriented in similar manner to each other in relation to the world coordinate system.

Designed parts for the simulation will be extremely simplified. In the real-world situation the tool stand would have some notches for the tools to rest in place. In the simulation, I have just put the tools in the tool stand to look visually nice. The work area is also different in the real-world situation because the tools might be put in different locations and the rotations are around a different axis.

Designed part for the work area has dimensions of 2 m width and 2 m height. It is positioned vertically. It will look like a wall. In the centre of this wall object there is an overhanging part that has dimensions of 0.5 m width and 0.5 m height. It is 5 cm outwards from the wall. At the centre of this overhanging section of the part there will be holes for those adapters that are in the tool stand waiting for installation. Movements that require rotational aspects will have a square hole. Work area will be moved closer to the robot with a platform. In the figure below there is the simplified work area design that I used in my simulation.

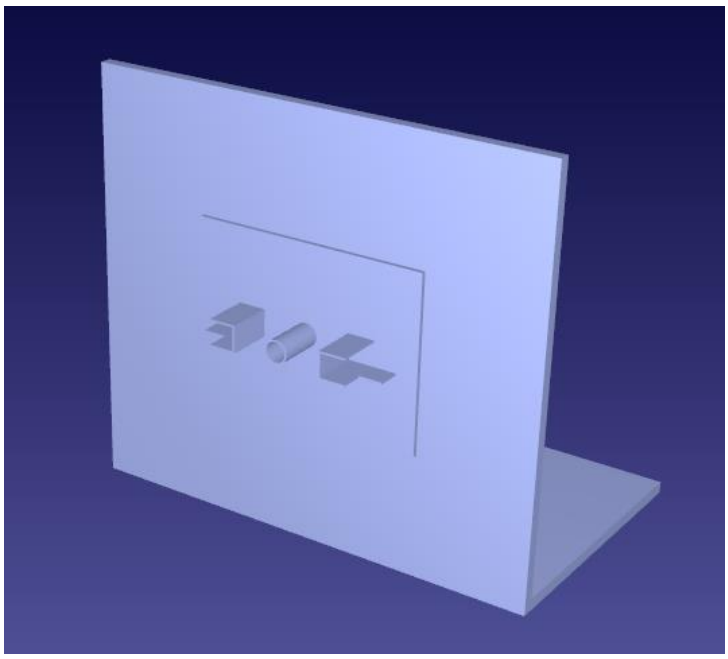

Figure 5. Simplified work area design

The tool stand has as many openings as there are tools in the simulation. It is clearly visible that in the tool stand that is in the figure below are not any notches for the tools to rest against. I placed the tools in the air in the simulation. Visualization was more important for me than to make the simulation as close as possible to an equivalent real-world situation.

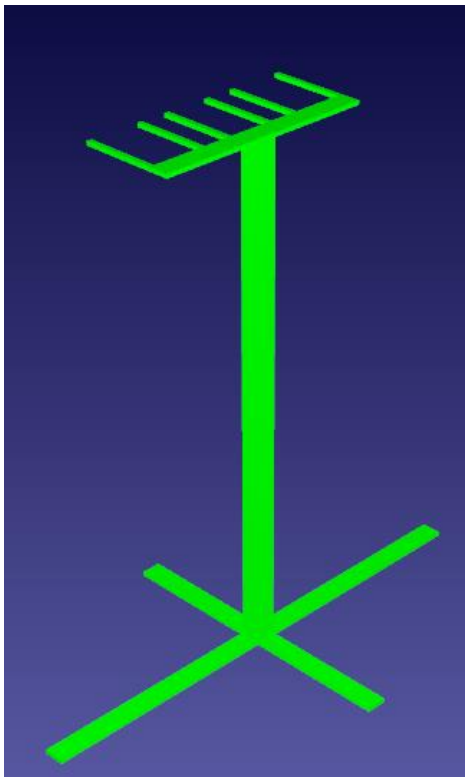

Figure 6. The tool stand

4.5.3 Simulation

Simulation will have many programs. There will be a single movement of the end-effector between two target points inside single program. Making a so-called main program is mandatory to make the movements continuous.

The hardest part of the simulation is to make sure collisions and singularities will not happen. This I found out to be slow manual work. As previously mentioned, to make sure that simulation will run without singularities then you must design enough target points. To avoid collision, I found the simplest way is just to visually confirm every movement that is close to other objects. In RoboDK there is a collision detection map, but it is not easy to understand how it works. I found that my tools would go through other objects. This means for me that the collision detection system is not completely functional in RoboDK just yet.

4.5.4 Reconfiguring workspace

Selected platform mechanisms only have one possible axis of movement. A linear movement program will make this movement. Making sure that right tool frame and robot is selected for this is the most important part of these programs.

Teaching start and end positions are the main thing for these linear movements. Simulation will end when the work area and the robot have parted away from each other. Tools should still be attached to the work area for the simulation to be completed correctly.

Parent-child combinations are also important in the workspace. In the simulation there are two parents, which are the linear movement mechanisms. Children are the robot manipulator and the work area wall. The robot will be a child for the KUKA KL 1000 2 linear mechanism, but it will also be a parent for the tool changer master part.

4.6 Exporting results

Benefits of exporting the simulation are that you can change angles and zoom in and out in the exported file. In RoboDK 3D simulation can be exported in two ways. Exporting saves the document in one file, either HTML or PDF. PDF is readable only in Adobe Acrobat Reader. PDF always also asks if you are sure that you trust the file. This means that you must allow Acrobat software to read the 3D-PDF files every time.

I will try both, but most likely will end up using HTML as an export choice. For me it is strange that you cannot make an animation video straight from the software. If you really want to get the simulation in video format, then you need to use some screen capture software to do that.

Exporting results are limited in the trial licence. You can export the simulation in PDF format, but with that there are problems. Simulation will look graphically strange and have glitches and not every object is visible in the simulation. PDF-export is not a very good exporting option. HTML is much better solution for the export, but without premium licence this is impossible to do. After testing both methods, I concluded that the HTML export is a much better option for exporting simulations, for just the fact that graphics work more smoothly in that.

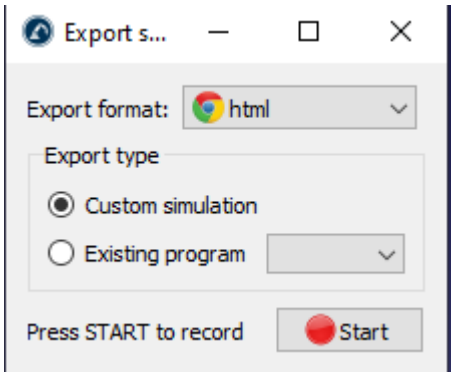

Figure 7. Exporting menu

The preferred exporting format is HTML. Export type is the desired program you want. A Start button starts the recording and asks you a place to save the exported file.

5 SOLUTION

The solution and the most important factors in it will be explained as accurately as possible. No singularities happened during the simulation. I am not sure that everything went fine in the simulation with contacts and collisions. I tried to avoid collisions by visually confirming every movement in the simulation. Singularities and the reach of the robot are the main aspects that you should always check.

5.1 Designing simulation in RoboDK

The selected manipulator has a reach of 1420 mm. I put every target inside the second outermost sphere in visible workspace in the simulation software. These spheres will be explained later in Chapter 5.1.4. This decision was made, because this way I can make sure that joint 5 made most of the installation movements. The selected robot manipulator for this simulation is KUKA KR10 R1420.

To show objects getting closer, I found out that easiest solution is to select the moving mechanisms from RoboDK library. These moving mechanisms are under the ext. axis. objects section on the library. I selected these moving platforms from the same manufacturer as the robot. This way the simulation will have similar colour scheme overall. I selected KUKA KL 1000 2 Base for the manipulator and for the work area I selected KUKA KL 3000. In the figure below the grey platform is the KUKA KL 1000 2 and the black one is the KUKA KL 3000. Both mechanisms have only 1 moving axis. KL 1000 2 has a reach of 1200 mm and KL 3000 has 3300 mm reach. I raised this KUKA KL 1000 2 mechanism 300 mm up and now the platform mechanisms are almost on the same level.

Figure 8. Placement of platforms

I have designed the following parts for this simulation: the work area in which the robot does the installation routine, a couple of different tools that are installed in the work area and the tool stand. Tools will rest in the tool stand. At first tool stand was not tall enough for this simulation, so I fixed that. The tools were designed in that way that those needing rotation at the end were square. This is because this way joint 5 makes the rotation not joint 6. I made the designs in SolidWorks 2020 software and saved them in STEP-format. Simplicity was a key factor in the designs.

I selected the tool changer from RoboDK library. The tool changer reference frame for the master part of the tool changers is at correct orientation. The hardest part with tool changer slave part was that it was not easy to determine which way was up for the tool changer. I ended up with a decision that every Cartesian coordinate system should be in the same direction and z-axis pointing towards the end of the designed tool. The tool changer slave parts were rotated around z axis in relation to the master part because this way the tool changer will look consistent in the simulation. I moved the TCP to the tip of the tool 1. This dimension is the same for every tool. This is problematic situation as the length of the tools will be restricted with this dimension.

5.1.1 Simulation plan

1. Pick needed objects from library
    - KUKA KL 1000 2 Base
    - KUKA KL 3000 Base

- KUKA KR10 R1420
- Zimmer WWR50F-B Tool Changer (Master)
- Zimmer WWR50L-B Tool Changer (Slave)
    - Make 2 copies of this

2. Bring desired objects that are designed in CAD program into the RoboDK (remember the right file format)
    - Work area wall
    - Tools
    - Tool stand
3. Move bases close to the world origin
    - Make sure that the robot on top the smaller platform and the work area on top the larger one.
    - Move platform reference frames closer together manually. Leave the world origin in the middle and between these platforms.
4. Make parent-child connections
    - Work area wall and KUKA KL 3000 base
    - KUKA KR10 R1420 robot and KUKA KL 1000 2 Base
    - Zimmer WWR50F Tool Changer (Master) and KUKA KR 10 R1420
    - Zimmer WWR50F Tool Changer (Slave, 3 copies) and Tool 1, 2, 3 respectively: one Tool Changer for one Tool
5. Manually move tool and tool changers together
    - Make their movements locked with each other
    - Select good position for TCP
        - This will be the same for all tools
6. Start making targets for movements
    - Remember to switch the robot when moving bases
    - Use similar configurations for the manipulator in every target point
    - Be careful of singularities and collisions
7. Pick tool 1
    - This is where you can confirm that tool changers are similarly orientated to each other.
    - Attach-event when picking up the tool
        - Connection with the designed tool and not with the slave part.

8. Move tool 1 in place
    - Get tool 1 to a so-called rotation spot
        - Rotation spot is a spot from where is it smart to rotate the tool in
    - Rotation is linear movement in the simulation
        - Joint movement would lift the tool while rotating
9. Detach tool 1 and pick tool 2
    - Call a detach-event
    - Make sure that robot arm does not collide with work area when leaving the area.
    - Attach tool 2, be sure to select the right tool from the menu.
10. Move tool 2 in place
    - Longer tool compared with other tools
    - Make sure when pushing the tool in place that it will not go through the wall
11. Detach tool 2
    - Event program
12. Pick tool 3
    - Attach-event, again be sure to select right tool
13. Move tool 3 closer
    - Singularity-event prevention
14. Rotate tool 3
    - Linear motion program
15. Detach
16. Move the robot manipulator to home configuration
    - Collision and singularity check here
17. Attach all tools to work area wall
    - Make sure the parent-child connection exists
        - Tools need to stay in the work area
18. Move work area wall back using KUKA KL 3000 platform
    - Linear movement with the platform
19. Move the robot back with the KUKA KL 1000 2 platform
    - Linear motion
20. Simulation ENDS.

5.1.2 Moving workspace in the simulation

As I mentioned earlier, I selected two ready-made mechanisms for moving workspace in the simulation. Movements are linear. The linear movement program is MoveL for the target. MoveJ is a joint movement program between two targets. Linear movements can be done with joint movement programs if the mechanism only has one axis. The end-effector of the robot usually does the joint movement programs.

Working solution for these linear movements was that for the work area I selected the wall object as a tool frame and moved that. I selected the KUKA KL 1000 2 Base as the reference frame for the linear movement of the robot. It was important not to select a tool for this robot movement because selecting the Zimmer tool changer as a tool would have meant that movement would have stopped when the tool changer would have been in the desired end position for the robot, and this also means that robot would have not ended up in the desired final position. Making sure that the moving platform was also selected as the robot for the program was important.

Figure 9. Example of environmental change program

5.1.3 Tool starting positions

In the following images, there are the locations for each tool in the tool stand. I wrote down the numbers in here because when I made modifications to the simulation, it would be easy to place tools back in their starting locations. To use these numbers, I double-click on the intended object to move and manually write the numbers from the images. As I was using a KUKA robot, I selected the coordinate system for the locations from the drop-down menu for KUKA robots. Only difference between the manufacturers in is the rotational aspect of the object in the coordinate system.

Tool changer is a child of a tool. Tool changer is visually put on top of the tool. TCP is put at the end of the tool 1, which is 220 mm in z-axis direction of the tool. Tool changer will move with the tool. Attaching the tool to the tool changer (master) part is done with visually placing the tool changer in correct position.

As I have target locations correct in the simulation, I could also use them to move objects back in place. Moving objects with target points I must remember to attach and detach tools when necessary.

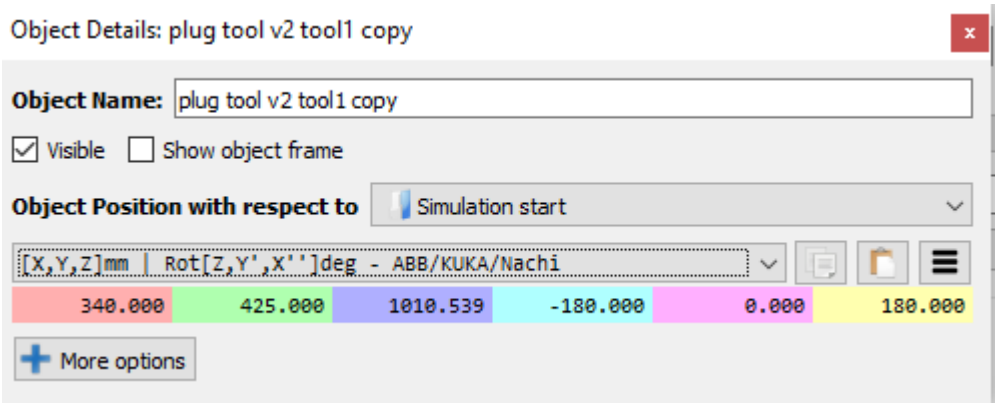

Figure 10. Tool 1 position in the tool stand

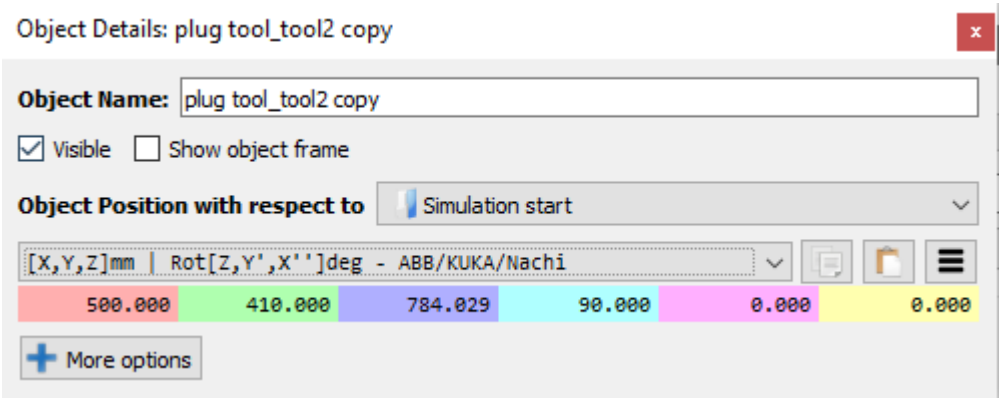

Figure 11. Tool 2 position in the tool stand

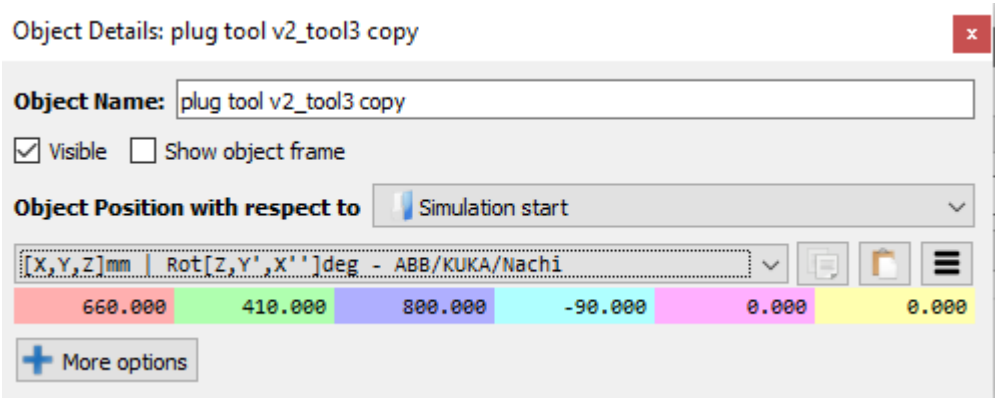

Figure 12. Tool 3 position in the tool stand

Even if you move objects back in the tool stand with attach and detach commands you should always check to be sure if locations are the same as in the above figures. Every simulation would have their own locations in the tool stand. I did not find a working solution to automatize this process. In my mind, you would write some Python code to do this. Python is the preferred code language in RoboDK. As this was not part of the desired simulation, I did not spend time to find a working solution for this problem.

5.1.4 Checking reachability

To see robotic manipulators maximum reachability, you can select the option for that to be visible from following location:

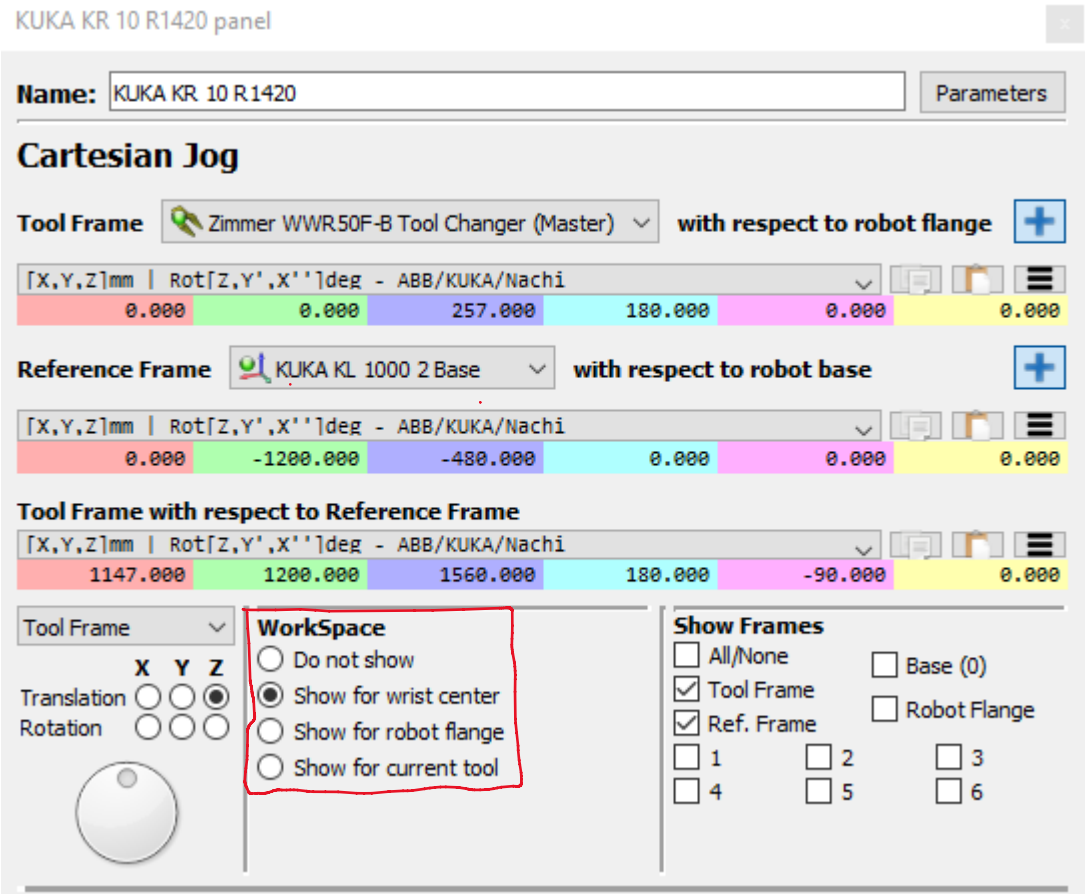

Figure 13. How to show workspace

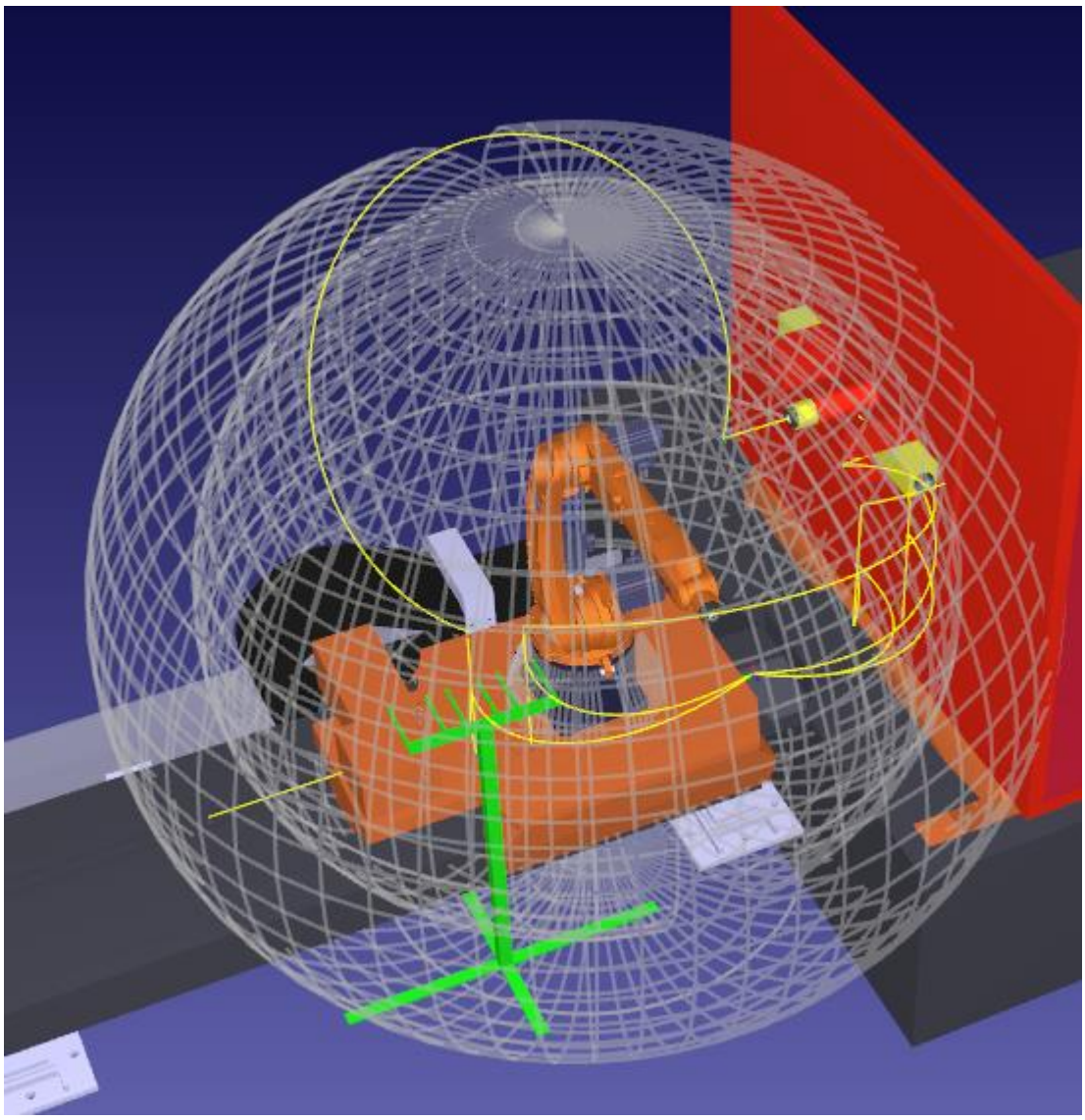

Figure 14. WorkSpace visible

To get to the KUKA KR 10 R1420 control panel you have a couple of options. Firstly, you can double-click the robot, or secondly you can select from the design tree menu the robot and right-clicking options from there. In the figure above workspace is shown for the wrist of the robot. Placement for the tool stand is not the best, but I selected that location, because now it was inside the second outermost sphere. I made this decision, because this way I can be sure that payload will not strain the robot arm as much as it would if the tool stand would be further away.

The tool stand is also placed slightly behind the robot because there is 10 degrees of so-called dead-space in the workspace. This dead-space is because joint 1 has the capability of movement ±170 degrees. This is not restricting other

joints. Placing the tool stand inside the area where joint 1 can turn to, makes movements slightly more fluid.

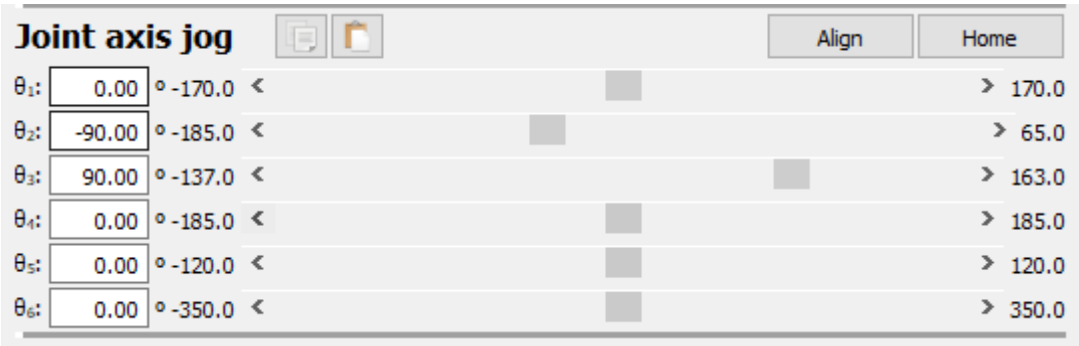

Figure 15. Joint angles in home position

In the figure above there are angles for the home position configuration for KUKA KR 10 R1420 manipulator. You will get this when pressing the Home-button from the location shown in the figure. The figure is a screen shot from the KUKA KR 10 R1420 control panel lower section. Double-clicking on the robot will open this panel. For this robot manipulator, the whole panel looks like as following:

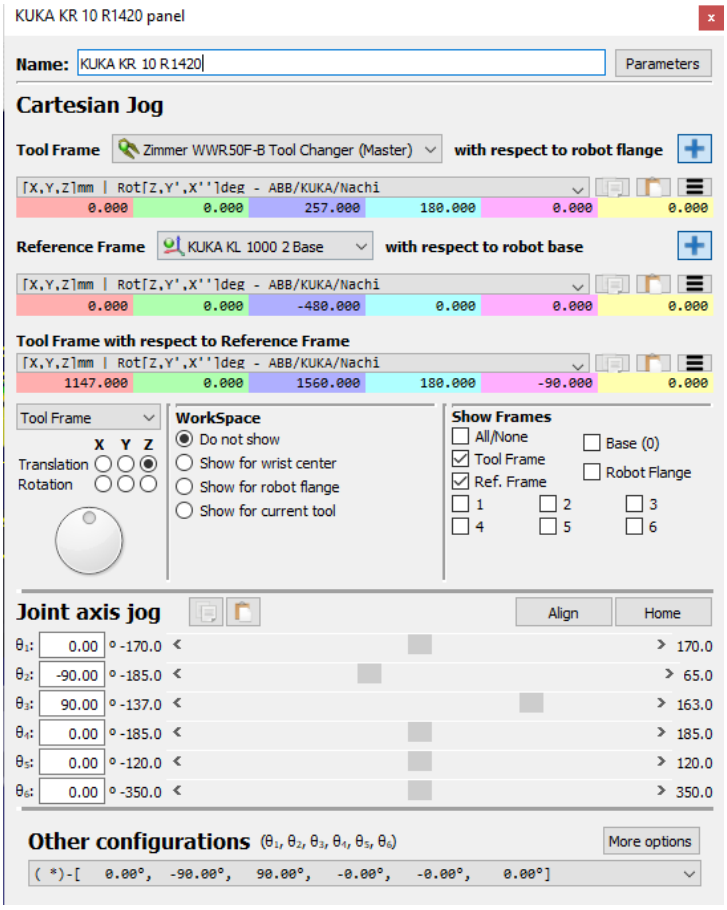

Figure 16. KUKA KR10 R1420 panel

5.1.5 Singularities

A screen capture of Options' menu in RoboDK is in the figure below. In this figure, it shows that tolerance to singularity is 2.5 degrees. If the end-effector of the robot goes inside this tolerance, then that joint will go the opposite configuration that it was originally in. This is while manually moving the robot arm. As an example, if in an original configuration elbow was up, then after singularity has been occurred then this same elbow would be down. If there is singularity in the movements, the program that has the singularity will stop the simulation.

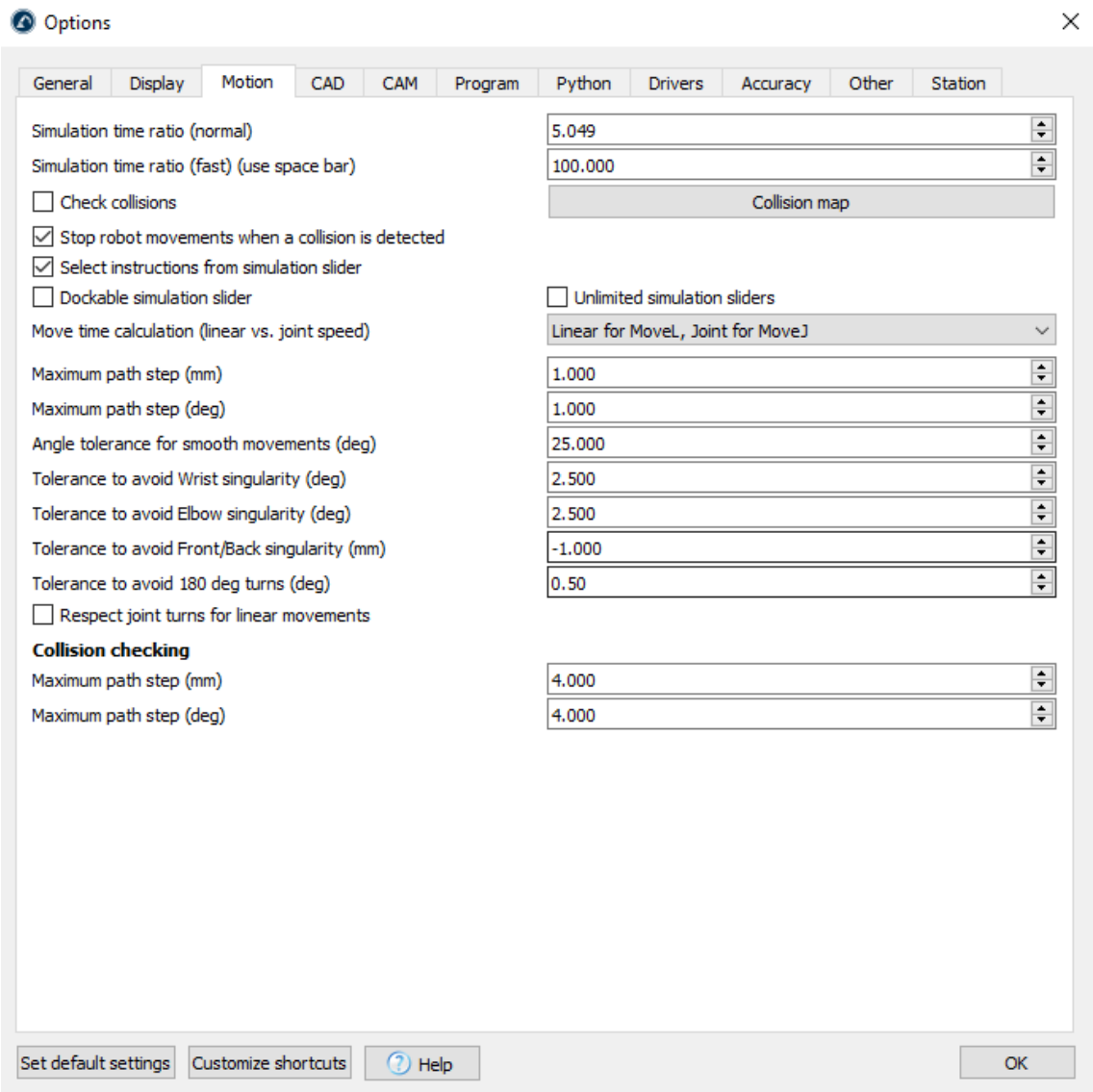

Figure 17. How to avoid singularities and collisions

The robot should stop its movements when a collision is detected. I have checked this box. As far as I understand this, this means that simulation would stop if robot were about to hit something. This did not work in the software. Objects went through other objects without warning. In conclusion, I would say that this part of the software is still underdeveloped.

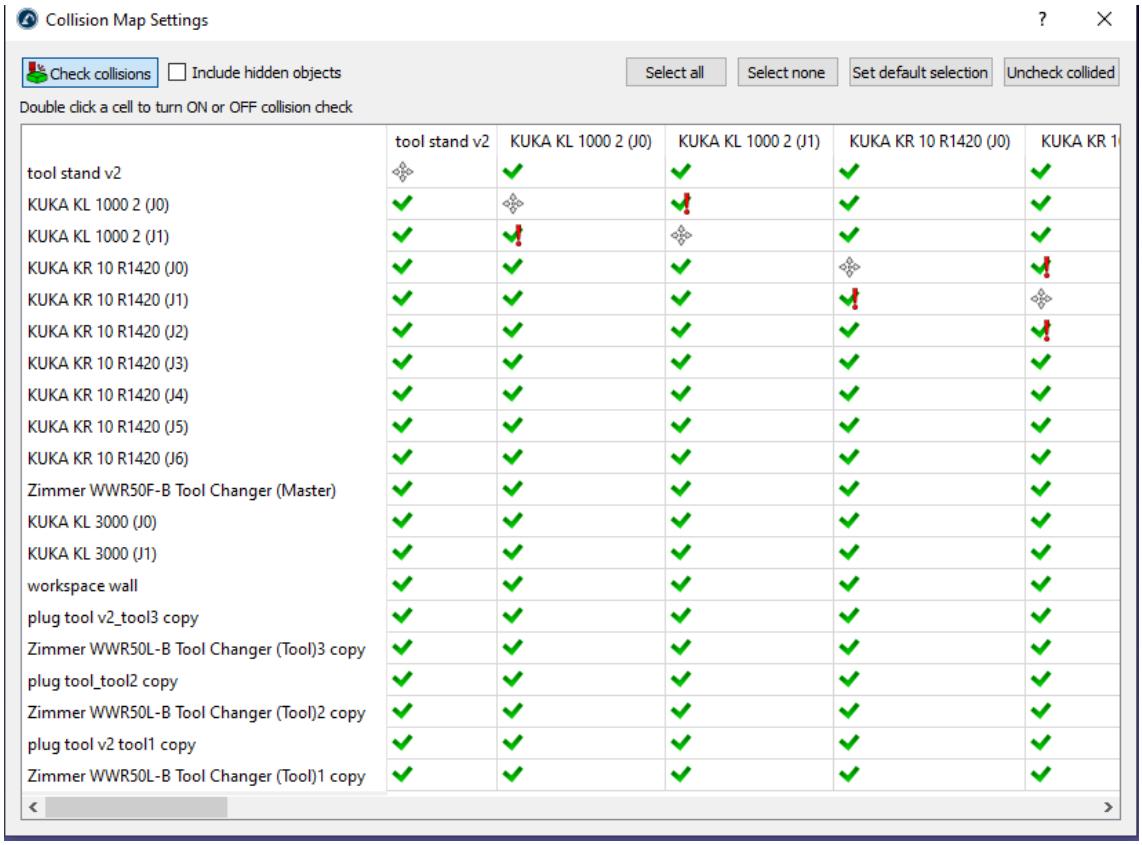

Figure 18. Sample of the collision map

5.1.6 Designing movements

Movements were designed to match the simulation plan written in Chapter 5.1.1. I made two kinds of movements in the simulation, linear and joint movements. Linear movements were used when lifting tools up from the tool stand and rotating the tools in place in the work area.

To make a movement with the robot you need two targets in the workspace, where from and where to. The where from target is the last location in the program, so that should automatically be correct. As I was moving tools in this simulation, I had to teach targets based on my tools. To get the target for the where to portion of the movement, you need to right-click the target in the design tree

and select options for the target. In the target options, you scroll coordinates with your mouse wheel. You can see a shadow of the robot while you scroll the coordinates. After scrolling the object to the right place, press the move to target-button and after that the teach current position-button. The buttons are shown in the figure below. This order is vital, because if this is done in the wrong order you must teach the target again.

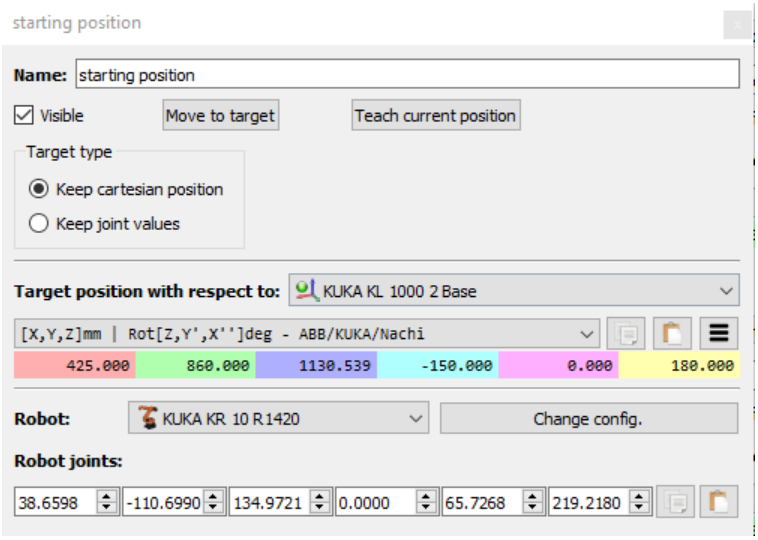

Figure 19. Example of a target options panel

The target position is with respect to the KUKA KL 1000 2 Base, because that mechanism is also a parent for the KUKA KR 10 R1420 robot. In my movements, I made sure that configuration with my robot is similar in every target.

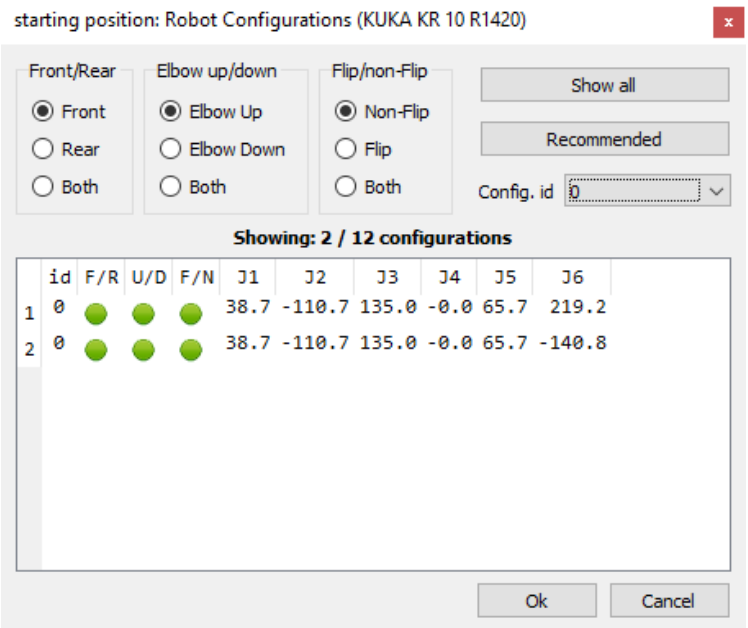

Figure 20. Robot Configurations menu

In the figure above is an example of the configuration I used in every target. The configuration id was always 0, which means based on the figure front, elbow up and non-flip, I also selected the first one of the possible configurations. To teach a configuration for the robot, I selected the wanted configuration and closed this robot configurations' menu from the OK button. Then in the target options part, I pressed the teach current position button again. This way joint angles in the robot are correct based on the wanted configuration. These configurations should be the same between target points because that way movements should look natural.

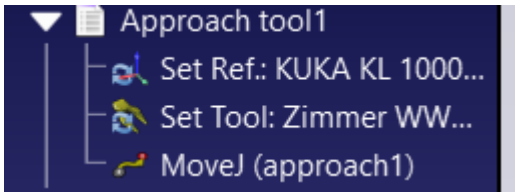

Figure 21. Example of a joint movement program

There is an example of a joint movement program in the figure above. In this program it is important that reference frame and tool frame are correct. The selected tool is the tool changer master part. To simulate multiple programs, I had to create a program that would call them. This is called the main program. Calling the programs in right order is mandatory for consistency reasons.

5.1.7 Attach and detach tools

You have to move the tool changer master and slave parts in a position where it is smart to attach them to each other. This is made with a movement program. The tools are attached in an event program. In the figure below, there is a solution how to attach a tool to the master part. Maximum distance was originally 200 mm. TCP for the tools was 220 mm as this is the length of the tools 1 and 3. This is the reason why I had to change the maximum distance parameter.

In the action section there is also an option to detach the object. This is an important event to do after placing the object in the work area because otherwise the attached tool would still be in action and then the simulation would look idiotic with the first tool always attached to the manipulator.

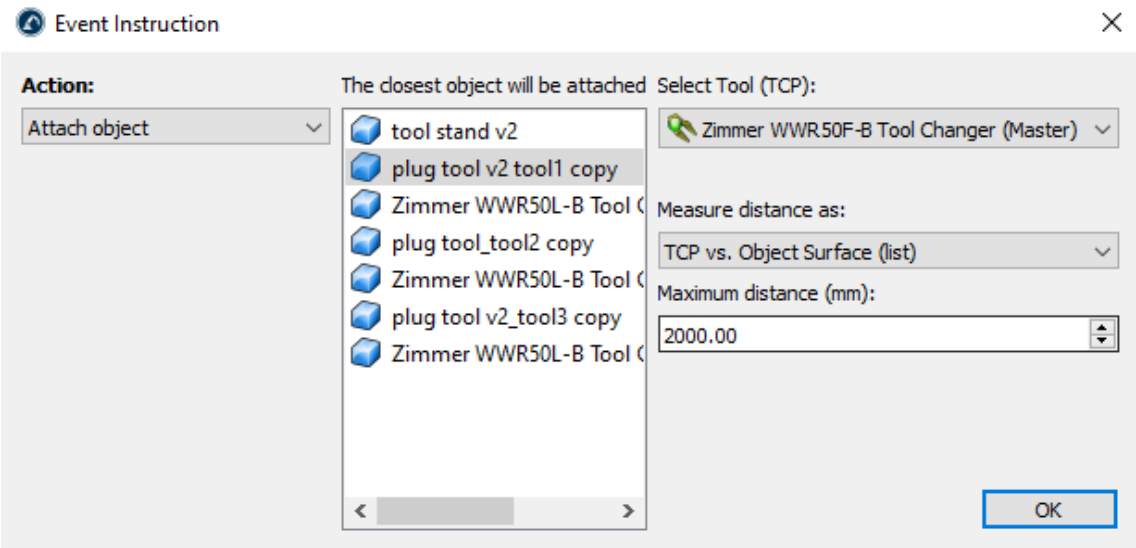

Figure 22. Event instruction program

6 DELIBERATION

The 30-day trial version was sufficient licence to get to this solution. A month was enough time to get know the basic functions of the software. The most difficult part of the simulation was to make sure that any singularities would not occur. Making sure that I had selected the right robot for the platform movements caused some head scratching moments because it is not said in the warning message that the selected robot was wrong. Usually in the warning message read that the reference frame or the selected tool were incorrect.

I started the thesis work by going through rather quickly the RoboDK documentation website. I also watched some YouTube-videos regarding the subject. The software maker has their own YouTube-channel. When I first thought that this thesis would be made using Visual Components software, I did the same work regarding that software. Comparing the software based on those YouTube-videos gave me the impression that RoboDK software developers had not figured out yet how to attach and detach different tools with the robot manipulator. Thankfully, I realized that I could do those attach and detach commands with the tool changers. That helped lot.

As the RoboDK main website page says no programming skills are needed in the simulation work so, this software could be a valid candidate when teachers are recommending software in robotics in the future. However, to work in the field of robotics you need to be a skilled programmer in at least one programming language. That is something that I have learned at Cargotec Corp. during my time.

SOURCES

RoboDK Pricing, webpage, referenced 30.10.2022. https://robodk.com/pricing

RoboDK Main Page, webpage, referenced 30.10.2022 https://robodk.com/

Automatic tool changers, RoboDK forum thread, Referenced 21.10.2022
https://robodk.com/forum/Thread-Automatic-tool-changers-in-RoboDK

Banks, J., Handbook of Simulation. Wiley & Sons. 1998.

Brailsford, S., Churilov, L., Dangerfield, B., Discrete-Event Simulation and System Dynamics for Management Decision Making. Wiley & Sons. 2014

Craig, J.J., Introduction to Robotics, Third Edition, Prentice-Hall, 2005

Forward kinematics. Referenced 24.10.2022.
https://en.wikipedia.org/wiki/Forward_kinematics

Z. Ying, S.S. Iyengar, Robot reachability problem: nonlinear optimization approach, Wiley, article published in March 1995

SCARA Robots, Fanuc, webpage, referenced 25.11.2022.
https://www.fanuc.eu/de/en/robots/robot-filter-page/scara-series

STANDARDS

ISO 10218-1 Robots and robotic devices – Safety standard for industrial robots Part 1: Robots
ISO 10218-2 Robots and robotic devices – Safety requirement for industrial robots, Part 2: Robot systems and integration