
FACULTY OF TECHNOLOGY

# DESIGN AND IMPLEMENTATION OF ROBOT SKILL PROGRAMMING AND CONTROL

Janne Saukkoriipi

DEGREE PROGRAMME OF MECHANICAL ENGINEERING

Master's Thesis 2019

# ABSTRACT

Design and implementation of robot skill programming and control

Janne Saukkoriipi

University of Oulu, Degree Programme of Mechanical Engineering

Master's thesis 2019, 83 p. + 1 p. Appendixes

Supervisor at University: University Lecturer, D.Sc. (Tech.) Toni Liedes

Skill-based approach has been represented as a solution to the raising complicity of robot programming and control. The skills rely heavily on the use of sensors integrating sensor perceptions and robot actions, which enable the robot to adapt to changes and uncertainties in the real world and operate autonomously. The aim of this thesis was to design and implement a programming concept for skill-based control of industrial robots.

At the theoretical part of this thesis, the industrial robot system is introduced as well as some basic concepts of robotics. This is followed by the introduction of different robot programming and 3D machine vision methods. At the last section of the theoretical part, the structure of skill-based programs is presented. In the experimental part, structure of the skills required for the "grinding with localization" -task are presented. The task includes skills such as global localization with 3D-depth sensor, scanning the object with 2D-profile scanner, precise localization of the object as well as two grinding skills: level surface grinding and straight seam grinding. Skills are programmed with an off-line programming tool and implemented in a robot cell, composed of a standard industrial robot with grinding tools, 3D-depth sensors and 2D-profile scanners.

The results show that global localization can be carried out with consumer class 3D-depth sensors and more accurate local localization with an industrial high-accuracy 2D-profile scanner attached to the robot's flange. The grinding experiments and tests were focused on finding suitable structures of the skill programs as well as to understand how the different parameters influence on the quality of the grinding.

*Keywords: industrial robot, robot programming, object localization, grinding*

# TIIVISTELMÄ

Robotin taitopohjaisten ohjelmien ohjelmointi ja testaus

Janne Saukkoriipi

Oulun yliopisto, Konetekniikan koulutusohjelma

Diplomityö 2019, 83 s. + 1 s. liitteitä

Työn ohjaaja yliopistolla: Yliopistonlehtori, TkT Toni Liedes

Robotin taitopohjaisia ohjelmia on esitetty ratkaisuksi robottien jatkuvasti monimutkaistuvaan ohjelmointiin. Taidot pohjautuvat erilaisten antureiden ja robotin toimintojen integroimiseen, joiden avulla robotti pystyy havainnoimaan muutokset reaalimaailmassa ja toimimaan autonomisesti. Tämän työn tavoitteena oli suunnitella ja toteuttaa taitopohjaisia ohjelmia teollisuusrobotille.

Aluksi työn teoriaosuudessa esitellään teollisuusrobottijärjestelmään kuuluvia osia ja muutamia robotiikan olennaisimpia käsitteitä. Sen jälkeen käydään läpi eri robotin ohjelmointitapoja ja eri 3D-konenäön toimintaperiaatteita. Teoriaosuuden lopussa esitellään taito-pohjaisten ohjelmien rakennetta. Käytännön osuudessa esitellään ”hionta paikoituksella” -tehtävän suoritukseen tarvittavien taitojen rakenne. Tehtävän vaatimia taitoja ovat muun muassa kappaleen globaalipaikoitus 3D-syvyyskameralla, kappaleen skannaus 2D-profiiliskannerilla, kappaleen tarkkapaikoitus ja kaksi eri hiontataitoa: tasomaisen pinnan ja suoran sauman hionta. Taidot ohjelmoidaan off-line ohjelmointityökalulla ja implementoidaan robottisoluun, joka muodostuu hiontatyökaluilla varustetusta teollisuusrobotista, 3D-kameroista ja 2D-profiiliskannereista.

Työn tuloksista selviää, että kappaleen globaalipaikoitus voidaan suorittaa kuluttajille suunnatuilla 3D-syvyyskameroilla ja kappaleen tarkempi lokaalipaikoitus robotin ranteeseen kiinnitetyllä teollisuuden käyttämillä 2D-profiiliskannereilla. Hiontojen kokeellisessa osuudessa etsitään ohjelmien oikeanlaista rakennetta sekä muodostetaan käsitys eri parametrien vaikutuksesta hionnan laatuun.

*Avainsanat: teollisuusrobotti, robotin ohjelmointi, kappaleen paikoitus, hionta*

# PREFACE

This master thesis was done with VTT Technical Research Centre of Finland between May and December of 2018. The work was a part of their project, where the possibility of grinding with industrial robot using off-line programming tool was investigated, designed and further developed.

I would like to thank everyone at VTT who have helped me with this thesis, especially Tapio Heikkilä for supervising and guiding the work and Tuomas Seppälä for all the assistance with the experiments. Also huge thanks to Toni Liedes for informing about this opportunity and supervising this thesis, as well as all-around excellent teaching attitude towards all the students in the University of Oulu.

In the end, I would like to give special thanks to my family and friends for all the support I have been given during the years. Without the excellent studying environment and your help, this would not have been possible.

Oulu, 30.1.2019

Janne Saukkoriipi

# TABLE OF CONTENTS

# ABBREVIATIONS

| | |
|---|---|
| API | Application Programming Interface |
| CAD | Computer Aided Design |
| CNC | Computerized Numerical Control |
| 3-DOF | Three Degrees of Freedom |
| 6-DOF | Six Degrees of Freedom |
| F/T | Force/Torque |
| I/O | Input/output |
| OLP | Off-line Programming |
| PCA | Principal Component Analysis |
| PCL | Point Cloud Library |
| PLC | Programmable Logic Computer |
| SME | Small and Medium-sized Enterprises |
| TCP | Tool center point |
| TCP/IP | Transmission Control Protocol/ Internet Protocol |
| TOF | Time of Flight |

# 1 INTRODUCTION

Robots were invented to do jobs that humans did not like of doing or could not do accurately enough. In the 1960s, humans were not able to manufacture products precisely enough for the industry, which led to the development of computerized numerical control (known as CNC) machines. Also at the time, there was growing use of radioactive materials that derived the need for a safe way to handle materials without exposing the human for the radiation. The early robots built in 1960s were outcome of the need of these two technologies, precise manufacturing and remote handling of radioactive materials. These robots had elementary control mechanics without any connection to the environment. (Siciliano & Khatib 2016, p. 2.)

The development of the integrated circuits, digital computers and significant size reduction of the electronic components made possible to design and program computer-controlled robots. The first industrial robots were then adopted by the automotive industry for flexible manufacturing systems in the 1970s. The reason for the automotive industry of being the first adapting industrial robots was the need for higher accuracy and a chance to cut the production costs. Welding was among the first tasks that robots were able to perform with better repeatability and precision than the human workers. After the welding had been automated, the robots started to took over other highly repeatability jobs such as painting, assembly and machining. (Siciliano & Khatib 2016, p. 2; Wallén 2008, p.12.)

After the automotive industry had proven their success with industrial robots, other industries such as the metal, chemical and electronics soon followed by (Siciliano & Khatib 2016, p. 2). Outside the industrial factories, new robotic applications for normal consumers took a few decades to enter the markets. From these applications, nowadays consumers are the most familiar with the vacuum cleaning and the lawn mowing robots.

As the robots have entered to the consumers market, industrial robots have been developed new ways to sense the real world. Nowadays industrial robots are able to perform much more than run the same sequence all over again. Machine vision techniques have been developed for the robots to locate the object autonomously. With 3D machine vision, it is possible for the robot to detect the object, identify it and perform operations to it without any human interaction. Force sensing sensors have been integrated to the robots, for example to allow them to “feel” the object before grasping it. Force control

can also be used for the machining processes, as for instance grinding can performed with optimal amount of normal force pointed to the surface of a working object to provide more even quality.

New ways to program the robots have also been developed, as the robots became more common. Traditional on-line programming was seen as time-consuming, difficult for complex tasks and less efficient because the robot cannot be used in manufacturing while the new program is being programmed. This led to the development of off-line programming. Off-line programming is done with computer software's that allows creating, modifying and simulating the paths without the presence of the actual robot. This way, downtime of the robot was cut significantly and programming made faster.

Even though the sensors have brought senses into robots and other programming methods have been developed, most programs are not easily applied to other tasks even by expert users. This has led to development of the concept of robot skills that are highly parametrized and easy to implement into processes even by non-expert users. The skills would rely into use of sensors and machine vision methods to provide more independent and comprehensive working methods. In the conceptual level of the skill-based programming, the operator would only need to give a few inputs, which would configure the program for different situations. For example, the user could only need to pick the type of operation and the working objects. The robot would use machine vision methods to locate the work objects and start performing the operations designated to it. This all leads to much more flexible production with less delay in the implementing process.

Main objectives of this thesis is to present the theory behind all the components needed for skill-based programs, design the skills required for grinding with localization -task and implement the skills to the industrial robot. Structure of the thesis will go as following. In chapter 2, all the elementary knowledge of robotics will be introduced. This will be followed by introducing industrial robot programming methods (Chapter 3) and 3D machine vision techniques (Chapter 4) that are used in the experiments. The concept and the structure of the robot skills will be explained in chapter 5. Chapters 6 will present the structure of the skills that will be implemented and the equipment that they are used in experiments. The results and conclusions will be presented in Chapter 7 and 8.

# 2 BASICS OF ROBOTICS

In this chapter, reader will be introduced to industrial robot system, common types of industrial robots and some essential topics of robotics that will come up when working with industrial robots.

## 2.1 Industrial robots

Industrial robot system at the most basic level will consist of manipulator, robot controller and pendant as can be seen in Figure 1. ISO 8373 (2012) standard gives a definition of a manipulating industrial robot as "an automatically controlled, reprogrammable, multipurpose, manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications."

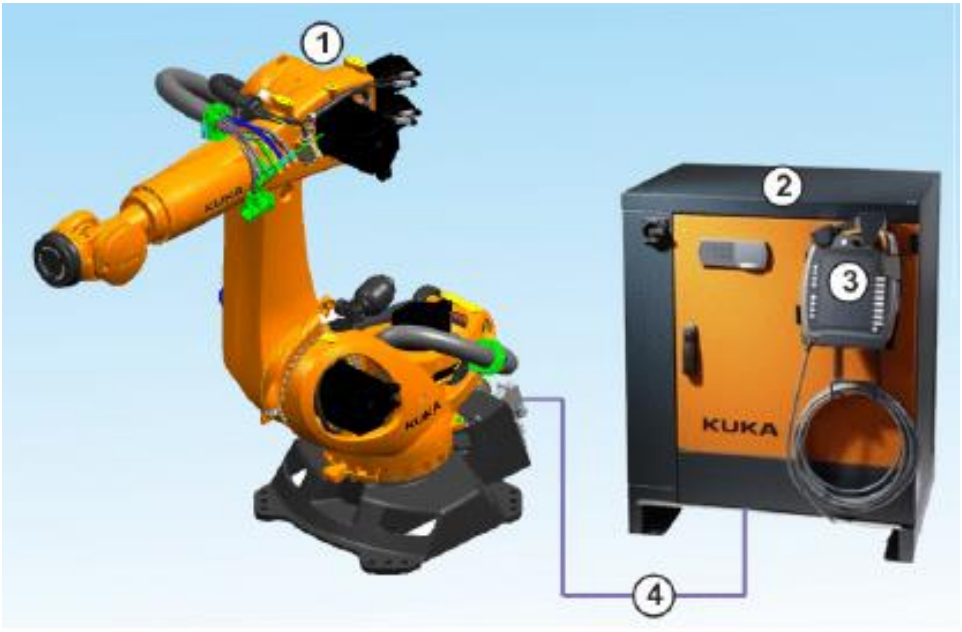


Figure 1. KUKA industrial robot system: manipulator (1), controller (2), pendant (3) and wiring (4). (KUKA 2015, p. 17)

All industrial robots come with pendant controller that allows manually controlling the robot. Without the pendant, the robot will need to be controlled straight from robot controller, which can operate in familiar computer operating systems (Windows, Linux). The pendant acts like a mobile controller that is connected to the main controller. It will also show the program in its own language and allow following the program from the code. This allows the robot programmer to get much closer to robot and visually check

the movements while running the program. This is useful in cases, where the programmer need to make sure that robots tool is in desired position relative to the working object. KUKA industrial robots use smartPAD, which can be seen in Figure 2.


Figure 2. KUKA smartPAD. (KUKA 2018d)

According to Siciliano & Khatib (2016, p. 1393), there are four main categories of industrial robots: Cartesian (Gantry), SCARA, articulate and parallel robots. Each of them have their unique workspace shapes and purpose of use. Generally, every robot uses electronic motors to control their joints with few possible exceptions. Cartesian, SCARA and articulate robot types will be shortly introduced due to similar physical structures.

### 2.1.1 Cartesian or Gantry

The most common applications for Cartesian robots are 3D printing and CNC-machines but they are very common in assembly lines. Since Cartesian robots have only three degrees of freedom (3-DOF) due to their three prismatic joints, their workspace shape is cuboid. Having 3-DOF limits their usability a bit, but there is usually no need for more in typical assembly lines. In most cases, their tasks include basic pick and place without any significant variation. One example of Cartesian robot used in palletizing is shown in Figure 3. (Siciliano & Khatib 2016, p. 1393.)

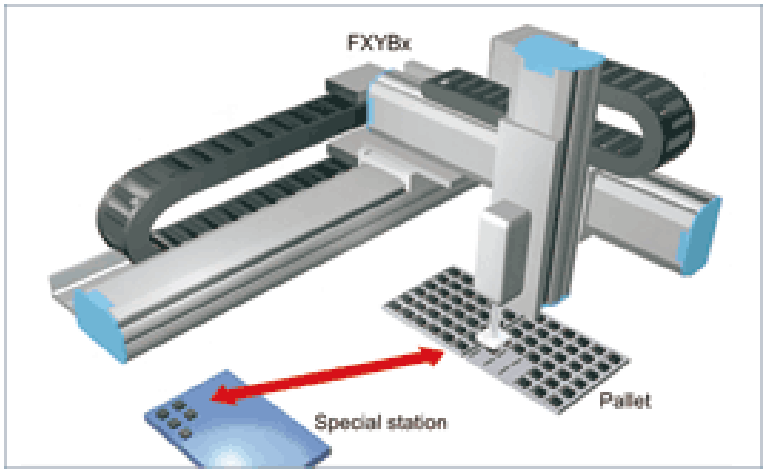


Figure 3. Yamaha's example of palletizing Cartesian robot. (Yamaha Motor Co. 2018)

### 2.1.2 SCARA

SCARA stands for Selective Compliance Articulated Robot Arm and it was designed and patented by Hiroshi Makino in 1981 to be more compliant for assembly purposes. By virtue of its joint layout, the arm is rigid in Z-direction but slightly compliant in X-Y directions. Compliance was seen advantageous in assembly operations such as inserting a round pin in round hole without binding. It has been proven faster and cleaner than Cartesian robot systems and still having a smaller footprint, which saves space in the factory. SCARA robot system and its workspace can be seen in Figure 4. (Makino 1982.)

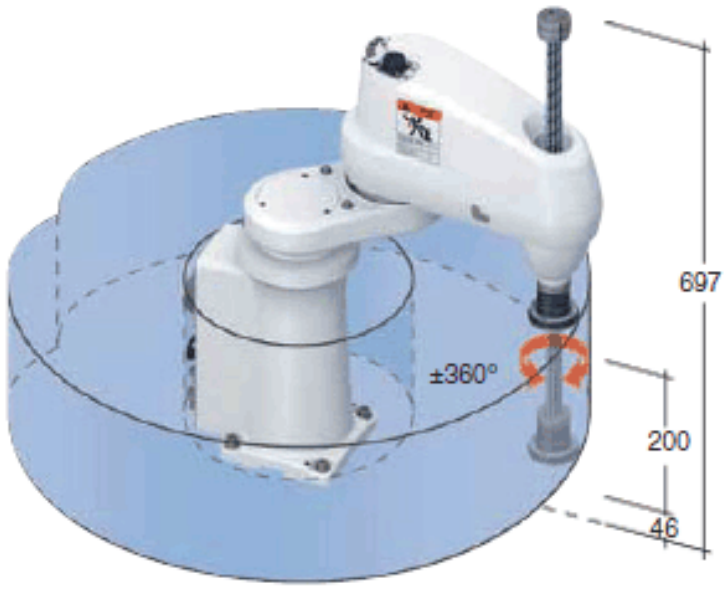


Figure 4. SCARA robot system and its visualized workspace. (Omron 2016)

### 2.1.3 Articulate robots

Articulated robot is made of multiple rotary joints and the number of joints can range from simple two-jointed systems for systems with ten or more joints (Siciliano & Khatib 2016, p. 1393). Number of joints required for the robot can be determined by analyzing how many degrees of freedom the robot needs for performing its tasks. For instance, 5-DOF robot has five degrees of freedom; it can move in X, Y and Z-directions and it can rotate around Y and Z-axes but not around X. In some cases, it might be enough to work efficiently.

The 5-DOF robot is still somehow limited to tasks that are more complex, which makes the 6-joint robot the more common articulate robot. 6-DOF has all of the six degrees of freedom that theoretically enables it to reach all locations and positions in its workspace. In practice, the robot's physical structure will limit the possible locations and positions. Figure 5 will show an example of 6-joint robot system. Robots, that have more than six joints, have several possible configurations for each end-effector position, which makes it easier to avoid singularities (explained later in 2.4) and other objects in range. Comparing to other robot types, articulate industrial robots have good reachability and high maximum payloads, which makes them very useful in the industry.

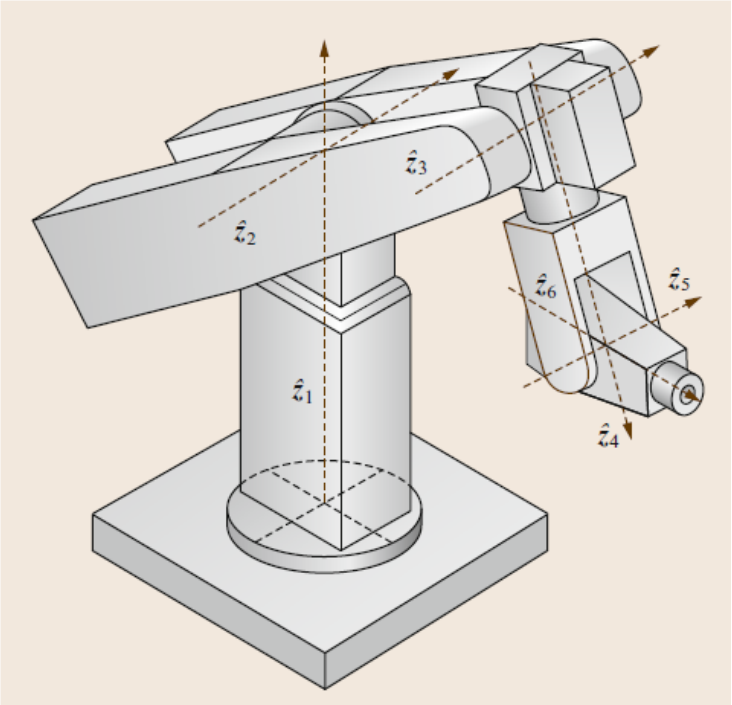


Figure 5. Robot with all six joint axes illustrated. (Siciliano & Khatib 2016, p. 26)

## 2.2 Coordinate systems

When programming industrial robots, the programmer will not manage to avoid working with multiple coordinate systems of which some examples can be seen in Figure 6. At minimum, a robot system includes at least three coordinate systems but usually there are many more. In addition to robots base coordinate system, there always is tool coordinate system and usually some kind of coordinate system for the work objects. Programmer can manually make new coordinate systems as needed and set their location in any given coordinate system. The robot will always have its own base coordinate system, which center will be located inside of robots root. The programmer can describe the location in any coordinate system and the robot controller will automatically use the rotation matrices to calculate the location in other coordinate systems.

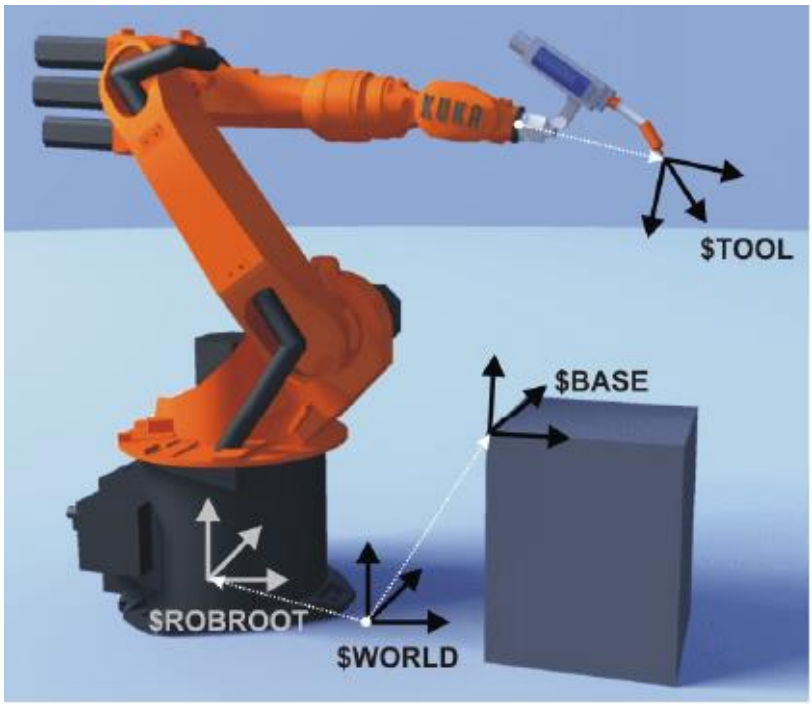


Figure 6. Different coordinate systems of a robot. (KUKA 2015, p. 64)

### 2.2.1 Joint coordinate system

Any possible position of a robot can be described by its joint values. For example, for robot with six degrees of freedom, it will show the angles of each of its joint relative to their reference axis. Its advantage is that while most robot positions in Cartesian system (explained in 2.2.2) have multiple possible joint configurations, but with joint coordinate system, the programmer can unambiguously determine the position of the robot. Joint

coordinate system also enables to programmer to use joint motions, which makes it easier to avoid crossing the robots singularities (explained later in 2.4).

### 2.2.2 Cartesian coordinate system

Cartesian coordinate system is a way to describe a vector in the world reference frame. It consists of coordinates (X, Y, Z) and angles corresponding rotation of each axes ($\Theta_X$, $\Theta_Y$, $\Theta_Z$). Together these six values will present the target which location and position is known. As for robots, it can for example represent the robot flanges reference frame location in robots base reference frame. X, Y and Z values are in metric or imperial units, which makes it easier for human to visualize the positions.

### 2.2.3 Tool center point

Tool center point (later known as TCP) is the point of a tool in which the robots location and orientation is defined (Figure 7). Robots perform the movements so that TCP reaches the target. A robot can have multiple TCP's in its memory but only one can be active at the time. It is important that TCP is configured accurately and calibrated properly. Without accurate TCP, for instance welding would not be done with the optimum distance or angle relative to the work object's surface. (KUKA 2015, p.125.)

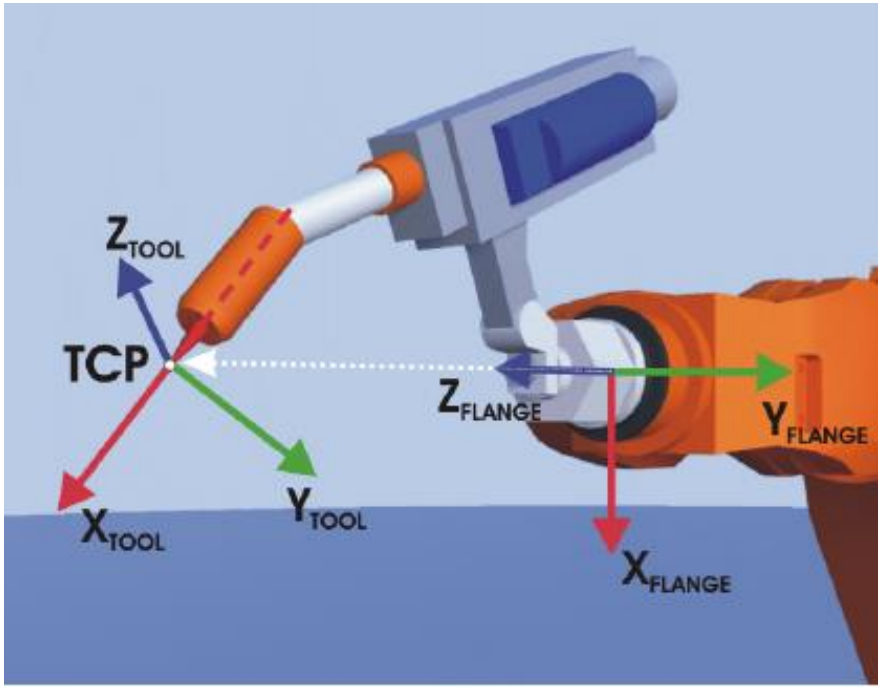


Figure 7. TCP location relative to the robot flange. (KUKA 2015, p. 125)

## 2.3 Kinematics

Kinematics is one the most essential part of robotics. When a location is expressed in Cartesian coordinates, robots controller needs to calculate the corresponding joints values so that the TCP reaches the target position. Transformation from joint values to Cartesian values is called forward kinematics and transformation from Cartesian position to joint values is called inverse kinematics (see Figure 8). For instance, because the linear motions are defined in Cartesian space, the robot would not be able to perform the motions without kinematics. Kinematics are also used to calculate the velocity or the acceleration of the end-effector in linear motions. (Siciliano & Khatib 2016, p. 28-31.)

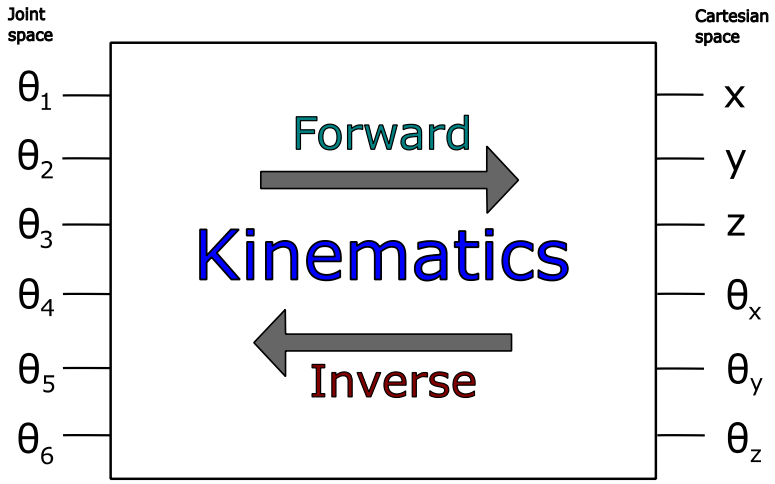


Figure 8. Difference of forward and inverse kinematics in 6-DOF robot.

Nowadays the controller of the robot will automatically use kinematic solutions to solve the required joint angles for the target positions, velocities and accelerations. Over the years, kinematics has been widely studied and lot of literature can be found on the subject. In this thesis, the problems and solutions in kinematics are not explained but more information can be found in books such as Siciliano & Khatib (2016) "Handbook of Robotics" or in Niku S. (2001) "Introduction to Robotics".

## 2.4 Robot configuration singularities

Adept Technology (2017) defines a configuration singularity as "location in the robot workspace where two or more joints no longer independently control the position and orientation of the tool". When a linear motion is executed near a configuration singularity, the robot joint speeds may become excessive while the robot tries to perform the motion.

An example of shoulder singularity can be seen on the left side in Figure 9, where the first axis of a robot will have to turn 180 degrees to perform the linear motion between the two targets. The faster the velocity of the linear motion, higher the angular velocity of the first axis needs to be. On the right is an example of wrist singularity, where the tool is nearly parallel to the robot's wrist. Because of the parallelism, holding the orientation of the tool while performing the linear motion between two targets is not executable. Due to the six-dimensional world, singularities are hard to visualize in figures. Thus, if better explanation is needed, videos illustrating the singularities can be found on the internet.

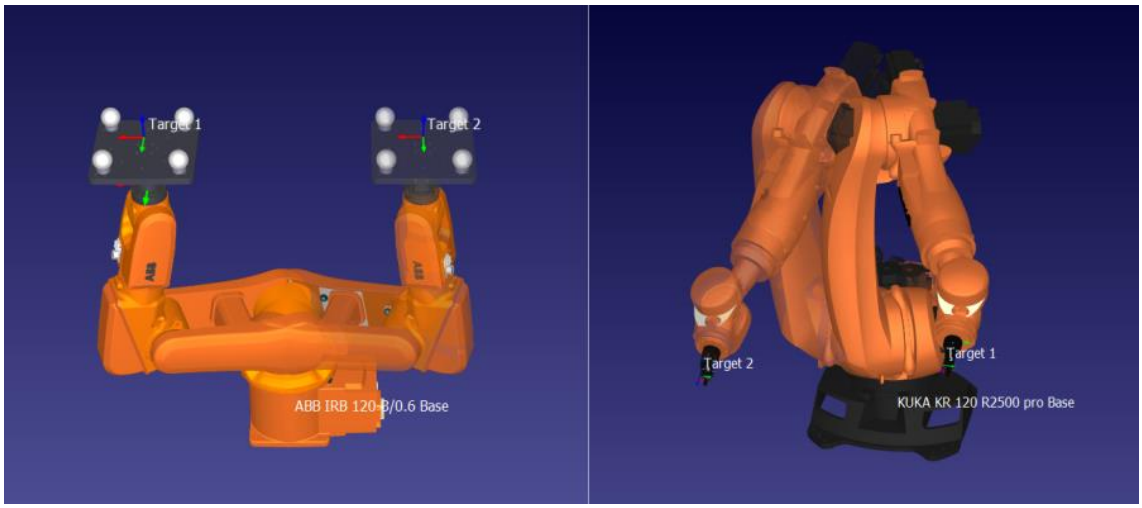


Figure 9. Two examples of robot singularities.

There are few ways to deal with configuration singularities. If possible, the movements could be moved to better area of the workspace where the singularities will not be close to movement paths. In the industry, this is not always an option since the work objects are usually very strictly located in the workspace and thus moving the objects significantly might not be possible. The other way to counter singularities is to use joint-interpolated move instead of a linear move. Once again, this way is highly dependent on the process itself. For example, welding usually requires many linear motions to ensure the right orientation of the tool. However, in tasks like pick-and-place, some movements can be completed by joint-moves without any functional differences. In the cases where the object cannot be moved and the linear motions are essential for the process, the robot with higher degree of freedoms is required for the task.

## 2.5 Accuracy and calibration procedures

When evaluating the performance characteristics of a robot, main parameters identified are the positioning accuracy, repeatability and resolution (Shiakolas et al 2015). These parameters are formed by the very structure of the robot such as the quality and capabilities of its main components, the controller, manipulator stiffness, and so on. Shiakolas et al (2015) defines the parameters as follows; resolution is defined as the smallest incremental move that the robot can physically generate, repeatability is a measure of the ability to return to the same position repeatedly and accuracy is defined as the ability of the robot to move precisely to the desired position in 3D space. Illustration of the difference between repeatability and accuracy can be seen in Figure 10.

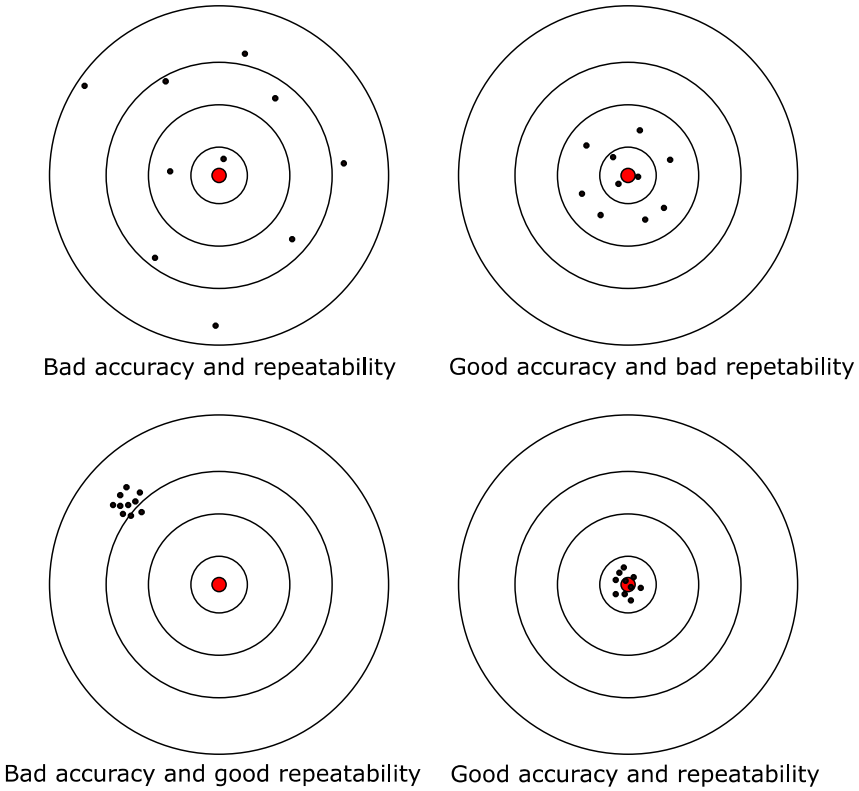


Figure 10. Difference between repeatability and accuracy.

### 2.5.1 Robot calibration

As the robots accuracy can be seen as vital for the robot's overall efficiency, different calibration methods have been developed to enhance robots positing accuracy. Calibration of the robot itself can be classified in three different ways: Level-1, Level-2 and Level-3. While Level-1 calibration will only model the differences between actual

and measured joint displacement values, Level-2 (also known as kinematic calibration) will consider entire geometric robot calibration including the angle offsets and joint lengths. Level-3 (also known as non-kinematic calibration) will also model other errors such as stiffness, friction and joint compliance. In most cases, Level-1 and Level-2 calibrations are efficient enough. (Elatta et al 2004; Everett et al 1987.)

### 2.5.2 Tool calibration for the robot

For the robot to use tools effectively, the tools need to be accurately calibrated. TCP needs to be known with certain accuracy that the robot can perform the actions as intended. Because the tool calibration is something that needs to be done for every tool in every robot, the robot manufacturers have integrated the calibration methods into their software. With KUKA, the TCP calibration is called XYZ 4-point method. The name comes from the procedure itself, as it requires the tool to be moved to a reference point from four different directions. The reference point itself can be chosen freely within robot's workspace. The robot controller will calculate the TCP from the different flange positions. The result of the calibration will provide the location and orientation of the TCP relative to the robot's flange. (KUKA 2015, p. 126.)

### 2.5.3 Calibrating machine vision sensors to the robot

For 2D-profile scanners and 3D-cameras used in this thesis (presented in 6.1) the used calibration procedure was originally represented by Tsai & Lenz (1989). It uses a stationary calibration block, which is scanned in five unique poses. To estimate the pose of the scanner, the planar 3D-model of the block is fitted the measured profiles of the laser scanner. Because the pose of the robot's flange is known in the robot base coordinates and calibration object is known in the sensor coordinate system, pose pairs are achieved. By observing the changes between the pose pairs in multiple set of poses, local pose-to-pose motions can be formed. These pose-to-pose motions share the same rotation axis that can be solved. After the rotation has been solved, the translation can be solved too. Illustrative figure of the procedure can be found in Figure 11. For the external sensors that are not attached to the robots flange, the sensor will remain static while the robot will rotate the calibration object. The mathematical principle stays the same. More information of the calibration procedures can be found from Ahola et al 2014. (Ahola et al 2014.)

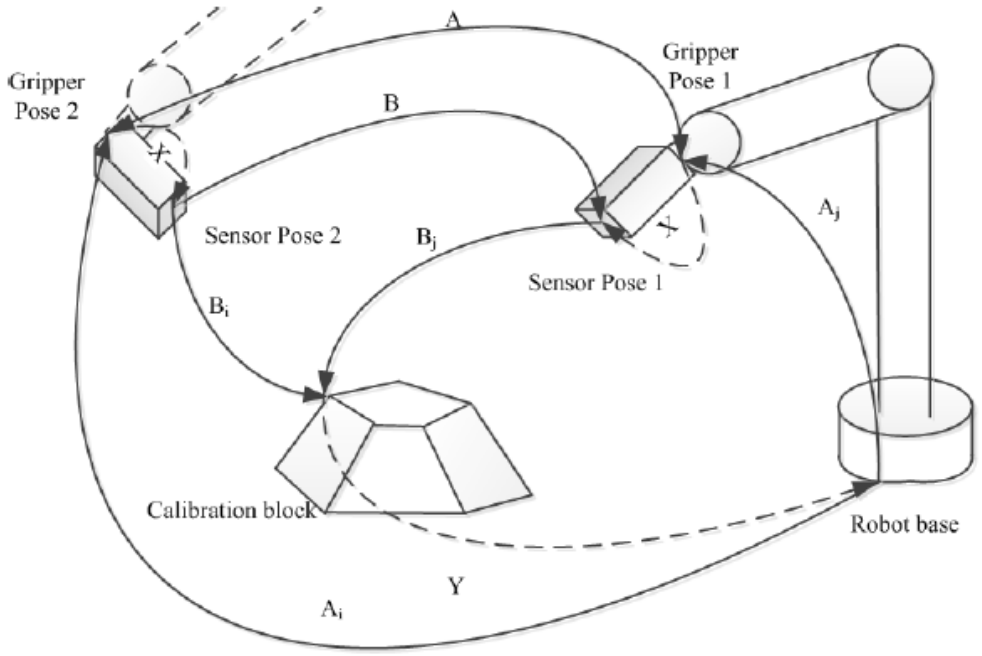


Figure 11. Calibration procedure. (Ahola et al 2014)

### 2.5.4 Calibration routine for force-controlled motions

Force-controlled motions are performed using a 6-DOF force-torque sensor that is attached to the mounting plate of the robot. Because the contact forces will occur between the tool and work object, the sensor is located between the robot and the tool to maximize accuracy. Force and torque signals of the sensor can be used for feedback control of the position and orientation of the tool. For accurately control the tool by the forces, the effect of gravitation force needs to be eliminated.

To eliminate the effect of gravitation, calibration procedure will be used to calculate the mass, center of gravity and orientation (in gravity field) of the tool. In the calibration routine, the robot will move the tool into random positions and record the force and torque values. The number of poses can range from a few (e.g. 4) to hundreds. Increasing the number of poses will lead to more accurate the calibration. After the calibration has been completed, the tool can be controlled by contact forces.

# 3 INDUSTRIAL ROBOT PROGRAMMING

In this chapter, the history of robot programming will be briefly explained which will be followed by the introduction of online and offline programming methods. In the end, RoboDK will be introduced as an offline programming software.

## 3.1 History of robot programming

The first robots were only used in teach-and-repeat mode, without the presence of any textual programming language. It was sufficient for applications, where a single task was repeated many times. However, there were number of factors that led to the development of a textual robot programming language. One of them was that teaching the path got more troublesome with rising complicity in run paths. Even a small change in trajectories would have required a new teaching routine. Another factor was that with textual language, the robot could be programmed offline and for multiple robots at once. The last significant factor was that it was harder to implement sensors for robots without any textual language in their use. (Gruver et al 1984.)

These days every robot program uses a textual language to represent the set of commands or instructions the robot performs. Programming languages are composed of words that distinctly describe the actions for the user. There have not been any standards for the languages, which has resulted a situation where every robot manufacturer has their own robot language. ABB has RAPID, Comau has PDL2 and KUKA uses KRL for programming the software application.

## 3.2 Online programming

Online programming with robot's pendant can be considered as a typical way to program robots. With textual programming language, a program can be easily made with pendant and later possible adjustments are easy to implement into program. With pendant, it is easy to follow the program and make small changes on the designated parts of it. New positions can be added into program by simply moving the robot into desired position and teaching the position for the robot.

Even though online programming is still a common way to program robots, it can be very time-consuming for processes that are more complex. Larger modifications into program can require a lot re-programming and repeating of the program with the actual robot. More modern day solution for robot programming are offline programming tools, which allow the programmer to program the movements and simulate them right away. The principle of off-line programming is presented next.

## 3.3 Off-line programming

Off-line programming (later known as OLP) is a robot programming method, in which a robot program is created without the presence of an actual robot as opposed to online programming in which the robot is programmed with teach pendant on site. After the program is finished, it will be uploaded to the robot for the execution. There are many OLP software in the market today, some are developed by the robot manufactures themselves and others are from third parties. Most of third party software support multiple robot languages, making them very useful in production lines that include robots from different manufactures. One major advantage from the industry perspective is the reduction of the robots downtime. With OLP, the programming and production can be carried out in parallel, rather than in series. This has cut the time of program implementation substantially. Also compared to traditional programming, programs that are generated offline are more flexible. Changes can be incorporated more quickly by only substituting the necessary parts of the program. (Pan et al 2011.)

In OLP's software's, a robot is represented by 3D-model in simulator, where the movement paths can be created, simulated and modified instantly. Movement paths can be simulated with accurate 3D-model of a robot. The simulation software will analyze the movement path and notify the user of any collisions, near-miss detections, reachability issues or joint singularities. The use of simulations decreases the chance of error and therefore improves the productivity and safety (Pan et al 2011). In Figure 12 can be seen a typical OLP software user interface, in which the robot is modelled accurately and movement path can be seen in yellow line.

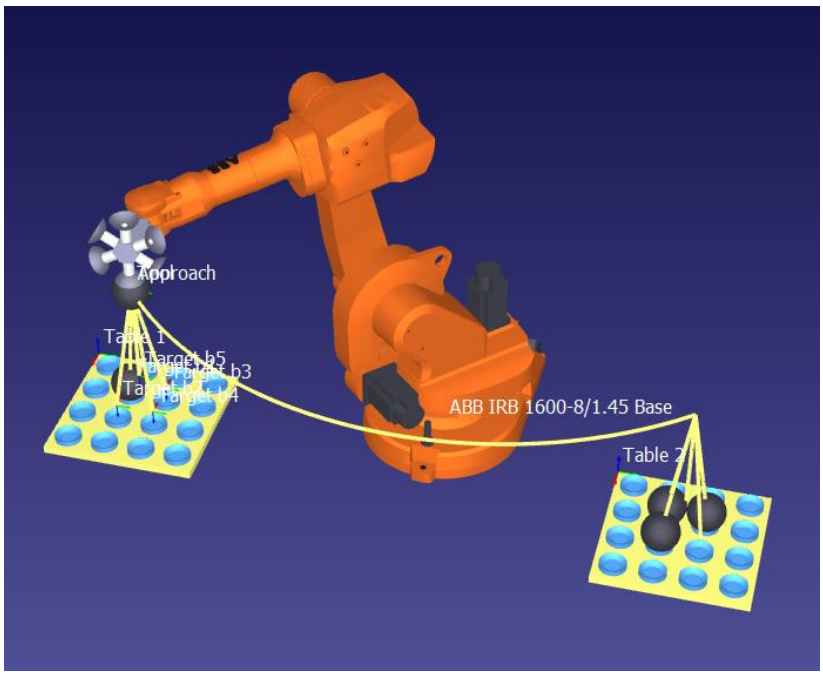


Figure 12. Off-line programming with RoboDK.

OLP software that supports multiple robot languages will have at least one post-processor for every supported language. The post-processor will do the conversion between the created program and the robot's unique language. It can be quite handy because one program can be easily translated to multiple languages and applied to various robots from the different manufacturers. (Pan et al 2011)

As promising as OLP's are, there are some obvious requirements and challenges for the efficient use. The first requirement for the utilization of OLP's are 3D-models of the work objects. Without them, programming accurate movement paths is impossible. This is not problem for the high volume industry, where 3D-models are standard in mechanical design processes. However, there are situations where there are no 3D-models available. In some small and medium enterprises (later known as SME's), low product volumes can be designed without the use of CAD. In addition, some manufacturing methods like plastic casting can produce products, which have a lot of variation between batches. In these cases, accurate movement paths cannot be produced off-line.

In addition, there are differences between the programs that has been generated off-line and the programs that have been programmed straight to the robot. Sensors can be burdensome to bring to simulations, because most of them need to be added into the simulation through application program interface (known as API, RoboDK example shown in 3.4.2). This makes it possible to create simulations of the whole process, but it

is very situational whether it is reasonable. Another issue that has come up with the use of post-processors, that when they generate programs, they tend to use the raw coordinates for everything. At least RoboDK's post-processors do not translate the loops, variables or for example, relative motion commands. It is safer and more reliable way to program but on the other hand, it makes the program's code much longer and a lot harder to read.

## 3.4 RoboDK

RoboDK is a third party OLP software for industrial robots and it uses an easy user interface, which enables basic robot programming for less experienced users. It has a library of over 300 robots for more than 30 manufacturers, which includes for example ABB, Fanuc, KUKA and Motoman. It can be seen as very beginner friendly platform for robot programming. 3D-model of a robot helps to visualize the robot positions and movements and making it easier to analyze the movement paths. RoboDK also shows the cycle time of the program, which is an advantage as in most manufacturing processes cycle times are limited. RoboDK allows importing 3D-models in STEP, STL and IGES formats, all of which are standardized and can be generated with any CAD software. For the implementation of sensors and actuators, RoboDK allows adding code lines to the program that have no influence in the simulation, but will actually give to actual robot all the necessary inputs and outputs that the sensors need. This allows using the sensors in the translated code that are not needed or used in the simulation. (RoboDK 2018.)

### 3.4.1 Post-processors

RoboDK currently has post-processors for all the biggest robot manufacturers such as ABB, Adept, Denso, Fanuc, KUKA, Kawasaki, Yamaha and so on. For KUKA robots, RoboDK offers all the seven kinds of KRL languages. It is also possible to add and modify post-processors in RoboDK as each post-processor is made with universal programming language Python. This allows the user to have more control to the generated robot-specific program. After the movements have been programmed in RoboDK, the program can be easily generated and inspected as can be seen in Figure 13. (RoboDK 2018.)

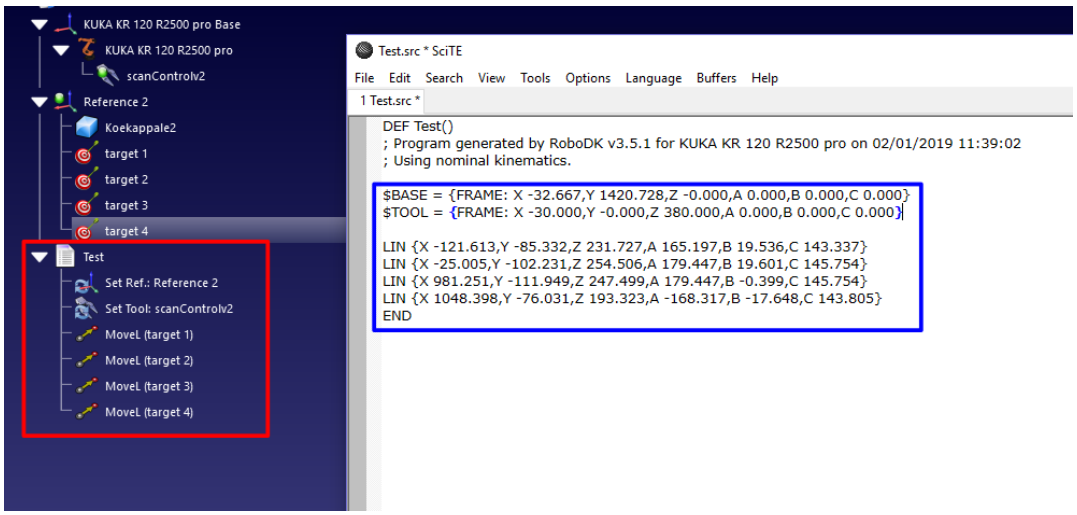


Figure 13. Generating program for KUKA with RoboDK. Red square shows the program in RoboDK and in blue is the generated program for KUKA.

### 3.4.2 Application Program Interface

The RoboDK API allows the user simulate and program for any robot using a universal programming languages such as Python, C#, C++, Visual Basic and MATLAB. API can be used to automate simulations and program the robot offline or online. Any of the supported programming languages can be used for creating macros that can automate specific tasks like for example, moving objects, reference frames or robots and simulate paint job by spraying paint in the work object. This way the sensors and actuators can be included in simulations. If the user is familiar with Python, it can be used to create the whole program, which is then translated to the robots unique language. With Python, new poses can be calculated from the others, which allows creating complex movements. In Figure 14 is an example, where the frames and targets have been set in RoboDK and the program itself has been programmed with the Python. Nevertheless, the same post-processor limits still exist. Even if the Python program used counting loops or conditional statements, those will not be included in the program that was generated. (RoboDK 2018.)

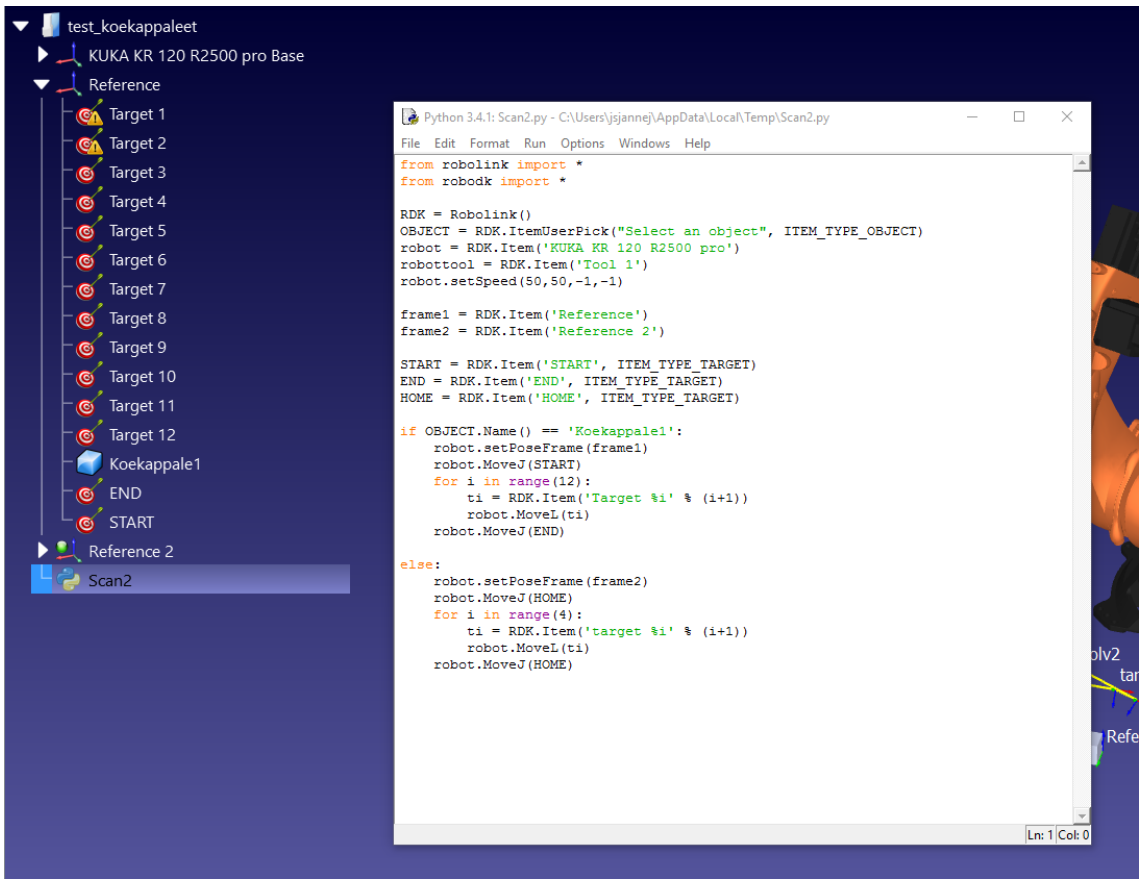


Figure 14. Python script programming from RoboDK API.

# 4 THREE DIMENSIONAL MACHINE VISION

Machine vision technologies can be divided into two main categories: 2D and 3D machine vision. Whereas 2D machine vision algorithms can be used to analyze regular images captured with regular cameras, 3D machine vision require the use of different kind of sensors. In this chapter, the structure of 3D machine vision systems will be explained, starting with the necessary equipment for machine vision and ending to techniques applied in this thesis.

## 4.1 Machine vision process

Beyerer J. et al (2016, p. 11) presented the structure of a typical machine vision system (see Figure 15) with five main phases. It begins with the image acquisition to capture the data, which is then digitalized and preprocessed. After the digitization and the preprocessing, the data should be in digital image form that can be segmented and the target features can be extracted. Target features can then be used in decision-making phase, which will eventually lead to desired action in the process. Next, the phases will be explained more thoroughly excluding the digitization phase, because it is depending and varying for each type of sensors.

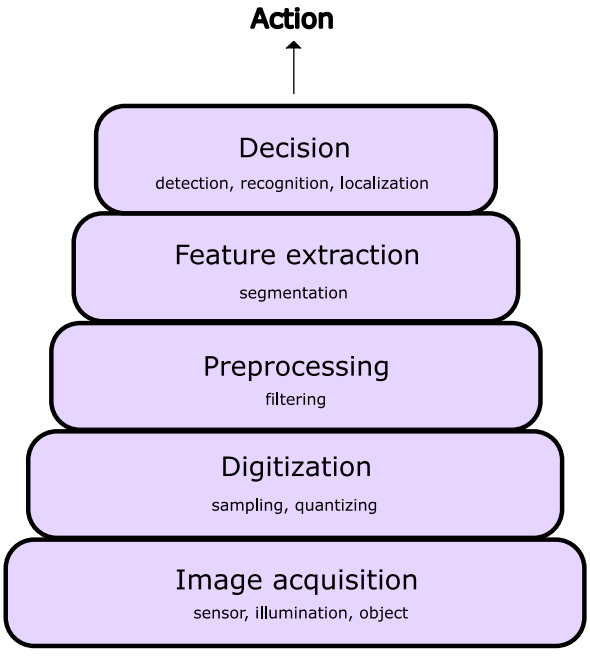


Figure 15. Typical the chain of process in machine vision systems. (Adapted from Beyerer J. et al 2016, p. 11)

## 4.2 Image acquisition

The first phase of the machine vision process is the image acquisition. All of the optical image acquisition and measurement methods rely on the properties of light. The light is directed through lenses to highly light-sensitive sensors with the shutter controlling the length of time of exposure to light. Same principle work even with human eyes, as they function as an organ that reacts to light. For humans, two eyes provide two images which together form a three dimensional image and each eye can differentiate millions of different colors. Even as the basic principle with eyes and cameras remains the same, machines are not necessarily limited only to visible light. Visible light is defined as the part of spectrum of light that is visible for human eyes (see Figure 16). For example, sensors can utilize infrared or ultraviolet parts of the spectrum if needed. (Beyerer et al 2016, p.23-24.)

In image acquisition process, light from the object is captured with cameras and its wavelength will specify the colors of the object. The main factors that define the quality of image acquisition are the optics and sensors used (what kind of camera is used), illumination (how illuminated is the object or the environment and shutter speed of the capture) and how reflective the object's surfaces are. (is the surface matte black or shiny metal).

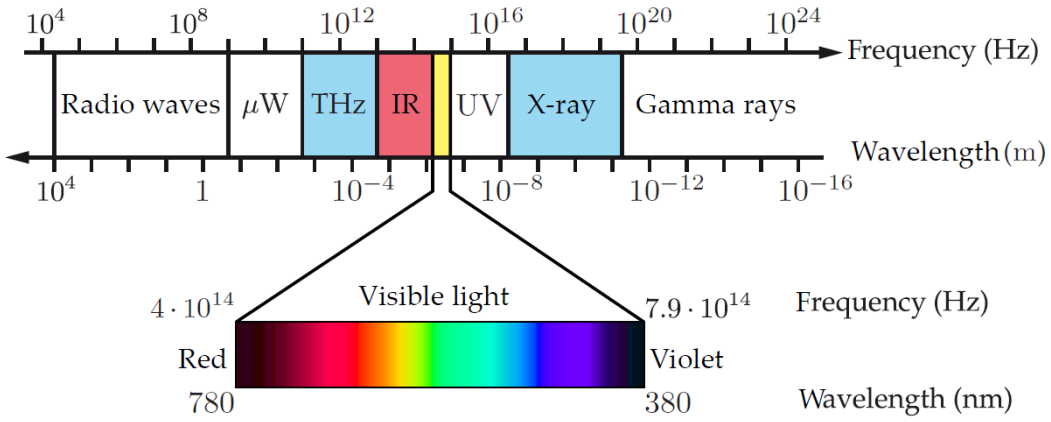


Figure 16. Electromagnetic spectrum of light. (Adapted from Beyerer et al 2016, p. 24.)

According to Perez et al (2016, p. 5), 3D vision techniques can be divided to passive or active vision methods and whether the image acquisition is done with single or multiple cameras. All the methods output a list of measured points - a 3D point cloud. Regardless of the method used to acquire the image, the result are the points that have three values

representing the X, Y and Z coordinates. Resulted point cloud needs to be then processed, like for example filtered and segmented into regions (segmentation explained later in 4.3). In Figure 17 can be seen an example of the point cloud. The arrangement of the data fluctuates with the method used to obtain it, some are arranged in order and some are not. It is important to know the arrangement of the measured data, because some algorithms used for processing can expect certain order of data.

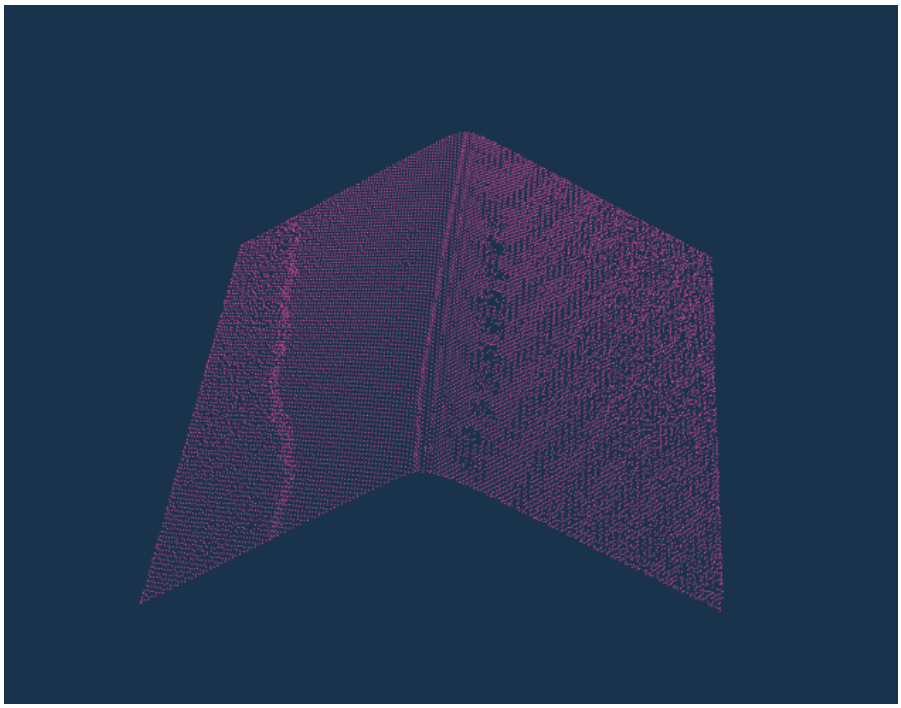

Figure 17. Point cloud obtained by the 2D-profile scanner.

The sensors used for experiments in this thesis utilize active vision techniques such as time of flight, stereo vision, structured light and laser triangulation, all of which will be explained next.

### 4.2.1 Time-of-Flight

Most active vision techniques project a visible or infrared pattern to the object to obtain 3D information, but the Time of Flight (later known as ToF) camera uses light pulses. The light pulse is usually generated by laser but also some special LEDs can generate such short pulses. The light pulse illuminates the scene after which the light will reflect from the surfaces, which is then captured by the camera. In addition, the flight can be modulated, and then the phase shift of the reflected light represents the range to the target (see e.g. Kinect 2 or IFM's sensors). Because the speed of light is known, delay of the reflected light pulse defines the distance to the object. The maximum range of the camera

is determined by width of the pulse and intensity of the illumination. Figure 18 will show the basic principle of ToF cameras. (Perez L. et al 2016, p. 5)

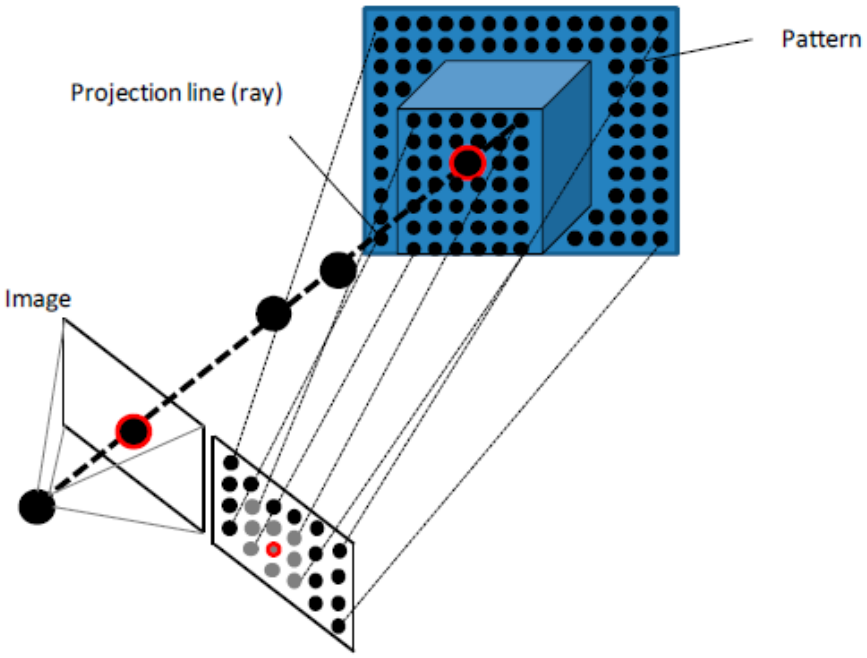


Figure 18. Time of Flight pattern projection. (Perez L. et al 2016, p. 7)

Even though the ToF sensors open new possibilities for industrial applications by being quite compact, lightweight and are able to operate almost in real-time, there are some limitations. These limitations include such as noisiness of the raw data, low resolution compared to other techniques, vulnerability of the distance errors and problems with multiple reflections. (Perez L. et al 2016, p. 8.)

### 4.2.2 Triangulation

Triangulation is a common process of determining the location of a point by measuring only angles to it from the known point locations. In 3D machine vision, it can be used to measure the distance between a sensor and point in an object. The principle of triangulation is applied in technologies such as structured light and laser triangulation. The difference between these technologies is that, structured light provides a 3D image of the scene where as laser triangulation outputs a line in 2D.

Coded structured light is based on projecting light pattern and viewing the illuminated scene from one or more points of view. Correspondence between the points of projected pattern and image points can be triangulated to obtain 3D information. There are many

different techniques for the structured light, but they can be divided into two groups: time-multiplexing and single frame techniques. In the time-multiplexing techniques, a sequence of binary or grey-scaled patterns are projected while the single frame technique projects one unique pattern. Illustration of structured light system can be seen in Figure 19. (Salvi et al 2004, p. 827; Perez L. et al 2016, p. 8.)

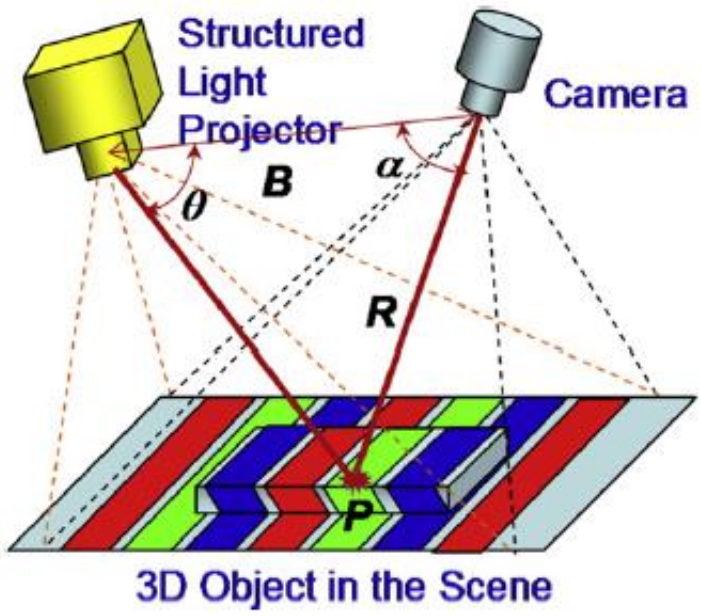


Figure 19. Illustration of structured light system. (Geng J. 2011, p. 131)

Figure 19 shows color-coded stripes as an example of the structured light. There are lot of different projection patterns available to use. Most common projections in the time-multiplexing methods are binary or gray coded and phase shifted patterns, or mix of both. Stripes and grid indexes are more used in the single frame methods. (Geng J. 2011, p. 133.)

Perez L. et al (2016, p. 8) also points out that the main constraint in multiplexing is that the object, the projector and the camera must all remain static during the projection. On the other hand, in the single frame methods moving the camera or the object is possible. However, Salvi J. (2004, p. 845-846) explains in his survey how different single frame methods still have limits with different patterns. Every pattern have their own advantages and drawbacks with the parameters such as resolution, accuracy, noisiness of the data and the depth per pixel.

Another technique that uses triangulation is called laser triangulation; a triangle is formed by the points of the camera, the laser emitter and the measured surface. Because the

distance between emitter and camera is known as well as the angle of the laser, the distance between sensor and measured point can be calculated by measuring only the angle of the measured point in respect to the camera. Illustration of the laser triangulation can be seen in Figure 20. (Beyerer J. 2016, p. 255.)

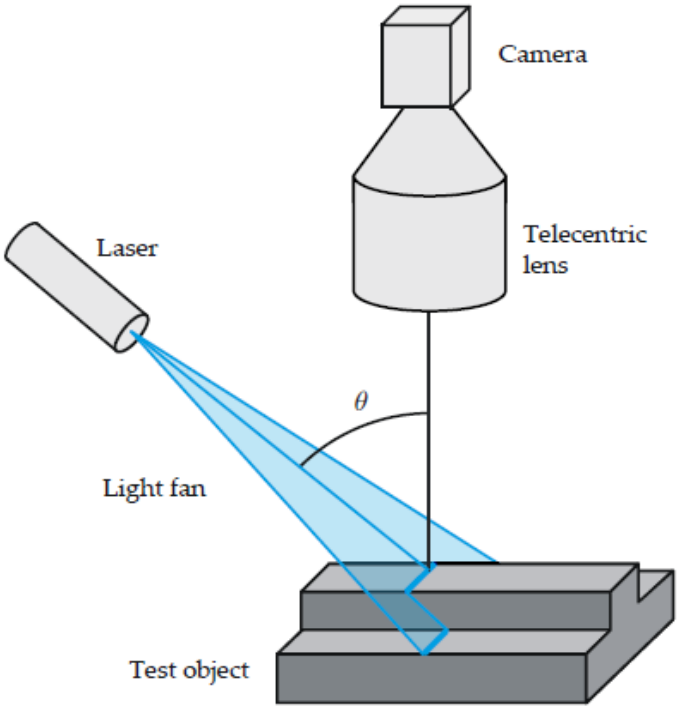


Figure 20. Working principle of laser triangulation. (Beyerer J. et al 2016, p. 259)

Non-contact laser techniques have developed considerably over the years. The early sensors gave isolated points on contrast to modern-day 2D sensors that measure across the dotted line while collecting multiple points at once. If the object or the sensor is moved, 3D model of the surface can be generated. These sensors can have very high accuracies as some can attain a measurement uncertainty approximately 1 µm (Beyerer J. 2016, p. 259). Due to high accuracy, line scanners have become more common in the quality inspections and in precise localization procedures. (Perez L. et al 2016, p. 10.)

However, there are some limitations for laser triangulation. Depending on the power of the laser, some can possess so high intensity that makes them unsafe for human eyes. In addition, the quality of the measurements can be hugely dependent of the surface material and quality. Especially mirror-like surfaces can cause problems with lasers due to higher level of reflection. Also comparing to other methods used to obtain 3D model of the surface, scanning time can be considered as a disadvantage. While the other methods

provide the image of the object in single frame, laser triangulation require sweeping motions to obtain full image of the surface. (Perez L. et al 2016, p. 10.)

### 4.2.3 Stereo vision

Stereo vision is a technique of estimating the 3D structure of the world based on two images taken from different viewpoints (Corke P. 2017, p. 479). Stereo pair is formed by two cameras, generally with parallel optical axes and separated by a known distance (see Figure 21). By comparing the two images provided by two cameras, the relative depth information can be obtained in the form of a disparity map. Disparity describes the difference in horizontal coordinates of corresponding image points and it decreases for points that are further from the camera. The relation of disparity to depth can be calculated for each system after which the depth image can be created. (Corke P. 2017, p. 483.)

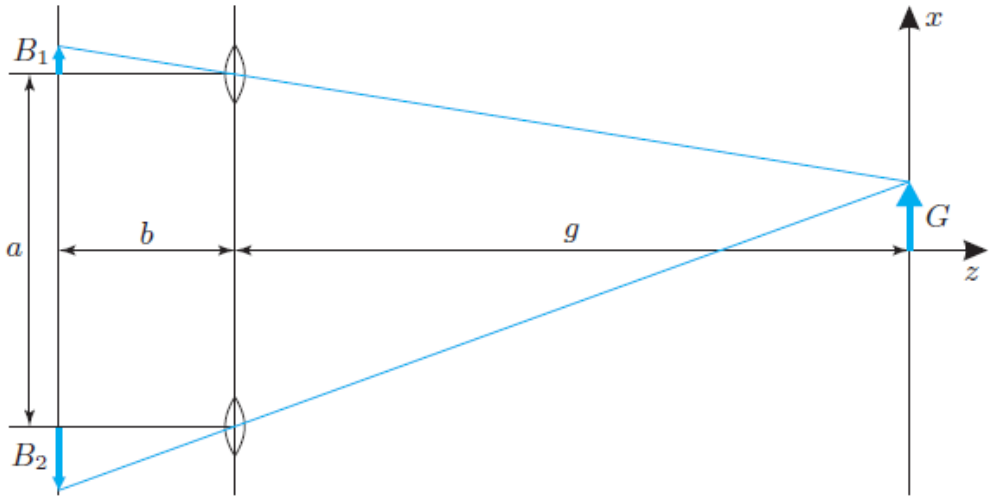


Figure 21. Working principle of stereo vision. (Beyerer J. et al 2016, p. 286)

According to Perez L. et al (2016, p. 7), stereo camera systems are widely used in many applications including indoor and outdoor robotics. They provide accurate depth estimates on well-textured scenes, but often fail when the objects contain low-textured or textureless surfaces. To improve the accuracy of low-texture surfaces, the scene is often pulsed with static infrared pattern to artificially create textures in the scene. For example, Intel RealSense D415 (presented in 6.1.4) has an optional infrared projector that will enhance the depth accuracy in scenes with low texture as seen in Figure 22.


)

Figure 22. Illustration of the infrared pattern used in RealSense D415.

## 4.3 Segmentation

In segmentation, measured point cloud is analyzed and divided into areas so that points in the same region share the same properties. According to Nguyen & Le (2011), the main challenges in the segmentation are high redundancy, uneven sampling density and lack of explicit structure of point cloud data. An example of segmentation can be seen in Figure 23, in which the same point cloud as showed in Figure 17 was segmented into two regions (green and cyan colored regions).

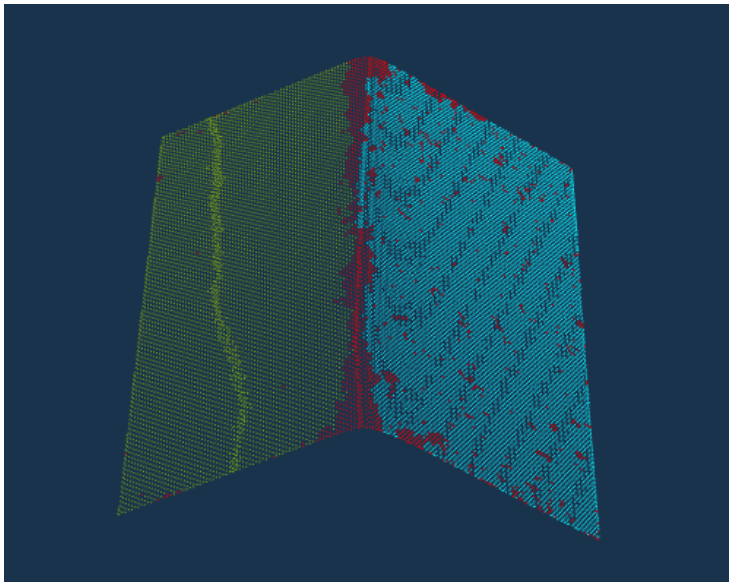

Figure 23. Segmented point-cloud with different colors representing planes except the red color represents dismissed points like edges.

Felzenswalb & Huttenlocher (2004) defined the requirements for segmentation methods that they believed to be the most important. First, segmentation needs to capture perceptually important regions or groupings, which often reflect the global aspects of the image. Secondly, the method should be highly efficient and able to run in real-time, which allows the technique to be used for video processing applications. Segmentation methods can be divided into five main categories (Nguyen & Le, 2011), which will be shortly presented next.

### 4.3.1 Edge detection

Shape of an object can be described by its edges. In edge-based methods, the principle is to locate the points, which have rapid change in intensity in 2D, or change in range or surface normal in 3D. Boundaries of the regions can be detected and used to obtain the segments. According to Nguyen & Le (2013), edge detection methods allow fast segmentation with drawbacks being low-accuracy due to high-level of sensitiveness to noise and problems with uneven density of point clouds. (Nguyen & Le, 2013.)

If noise of the signal is too high, the algorithm will detect edges that do not exist due to intensity changes. In addition, some lines can be completely missed due to same noisiness issue. Another problem is related to localization of the edge, as the noise can shift the position from its true location. Because of the sensitiveness to noise, filters have widely been used in edge detection methods. Even though filters suppress the noise level, it will also blur significant transitions. In Figure 24 can be seen an example of the edge detection. (Basu, 2002.)

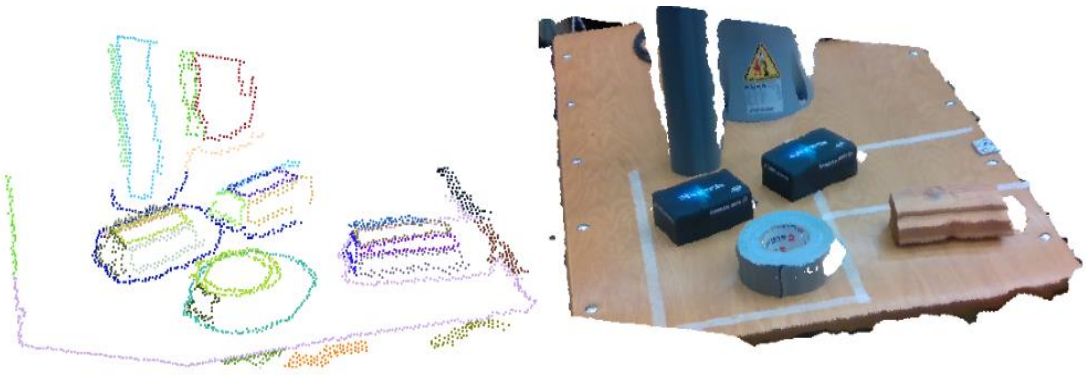

Figure 24. Edge detection segmentation.

### 4.3.2 Region-based methods

Region based methods rely on the neighborhood information, combining the nearby points that have similar properties into isolated regions. Consequently, it finds dissimilarities between the different regions. Compared to edge detection methods, region based methods are less sensitive to noise. However, the challenges are related to over or under segmentation and determining region borders with accuracy. Region based methods can be divided to top-down and bottom-up methods. (Nguyen & Le, 2013.)

Top-down and bottom-up region methods work in the opposite order, but with the same principle. In the bottom-up (also called seeded-region) method, the segmentation process will start by choosing a number of seed points. From these points, each region will grow by adding the neighbor points that satisfy the criterions set to it. The main challenges with method are sensitiveness to noise, time-consuming algorithms and dependency on selected seed points. Poorly chosen seed points can lead to under or over segmentation. The top-down (also called unseeded-region) methods will start by grouping all the points into one region. Then the algorithm will start dividing the region into smaller ones. The process will continue until the set thresholds are met. This method might have problem with over-segmentation when segmenting objects that are more complex. In addition, it is difficult to decide where and how to subdivide. It requires a lot of prior knowledge to configure the parameter values. (Nguyen & Le, 2013.)

### 4.3.3 Attributes-based methods

In attribute-based methods, the point cloud will be clustered based on the selected attributes. Method consists of two steps, the first step is attribute computation and the second is clustering based on the computed attributes. The method offers flexibility in the segmentation process by incorporating different attributes into it. The measured point cloud data can be defined with attributes such as color, distance, point density and horizontal and vertical point distribution. After computation, the height difference and the slopes of the normal vectors (explained later in 4.4.1) can be used as clustering attributes. A big advantage of this method is that it can eliminate the influence of noise or outliers. Limitations for the method include the dependency on the quality of derived attributes and it can be quite time-consuming. (Nguyen & Le, 2013.)

### 4.3.4 Model-based methods

Model-based methods use geometric primitive shapes to group the point cloud data. Geometric shapes include such as spheres, cones, planes and cylinders. All the points that have same mathematical representation will be grouped as one segment. With these methods, basic shapes can be automatically detected from unorganized point clouds. Some methods have been developed to detecting more complex primitives like for instance: helix, plane, and linear extrusions. The most known algorithms are RANSAC (Random Sample Consensus) and Hough transform. Both methods are used to detect the mathematical features. (Nguyen & Le, 2013.)

The RANSAC algorithm extracts the shapes by randomly drawing minimal sets from point cloud data. A minimal set is defined as the smallest set of points needed for uniquely to recognize a given type of geometric primitive. The primitives are then compared against all of the points in the data. After a given number of trials, the best candidate for the approximation of the points is then extracted and the algorithm will continue with remaining data. (Schnabel et al, 2007.)

The drawbacks of this method include the need for accurate scaling of the input point clouds and size of the shapes within the data. Another limitation is their inaccuracy when dealing with different point cloud sources. On the other hand, because model-based methods have purely mathematical principles, they are fast and efficient. In addition, they are robust with outliers or high-level of noise. (Nguyen & Le, 2013.)

### 4.3.5 Graph-based methods

In graph-based methods, edges are found with pairing the neighboring pixels and each edge will have corresponding weight factor. Weight is a non-negative measure of dissimilarity between neighboring pixels. The dissimilarity of the pixels can be measured by the color, difference in intensity, motion, location, angles between the surface normals or other local attribute. Edges within the same component should have low weight and edges between different components should have higher weights. (Felzenszwalb & Huttenlocher, 2003.)

According to Nguyen & Le (2013), when compared to other methods, graph-based methods can perform segmentation better in the complex images with noise or uneven

density of the point cloud. As a drawback, they mention that methods are usually not able to run in real-time. Examples of graph-based segmentation can be seen in Figure 25.

Figure 25. Graph-based segmentation. (Felzenszwalb & Huttenlocher 2003, p. 178)

## 4.4 Object localization

After segmentation has extracted desired features from the measured point cloud, results can be used for data analyzation, recognition or localization purposes. The phases can include actions such as estimating pose of the object, capturing features for recognition purposes or other data analyzation. The order of the phases can vary between different applications.

### 4.4.1 Pose estimation

A surface normal describe properties of the geometric surface. As an example, orientation of the plane can be described by the normal vector, which will be perpendicular to the surface. Surface normals may have been calculated locally (as seen in Figure 26) in segmentation phase. However, whereas the local surface normals describe the orientation of the points in a segment, principal component analysis (later known as PCA) can be used to characterize the whole segment. PCA is an efficient method to calculate variances for a point cloud, although the point cloud should be at least filtered to reduce the noise and other errors in the data.

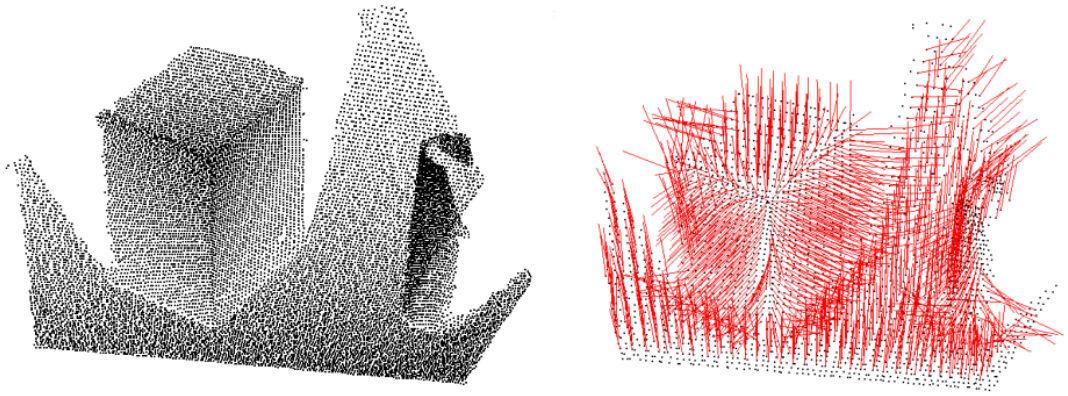

Figure 26. Raw 3D point cloud (left) and with visualized local surface normals (right).

PCA was explained thoroughly from Shlens (2014) in his survey that represents PCA as a standard tool in the modern data analysis because it is simple, non-parametric method for extracting relevant information from data sets. With PCA, a cloud of data points can be analyzed and variances for all directions can be calculated. When calculating the surface normal for a plane segment, the direction with the least variance represents the surface normal. As the other eigenvectors for a point cloud can be also calculated, they can be used to compute a coordinate system for an object. The starting point of the vectors is in the center of the point cloud. Figure 27 shows an example, where two eigenvectors have been determined for 2D point clouds. (Shlens 2014.)

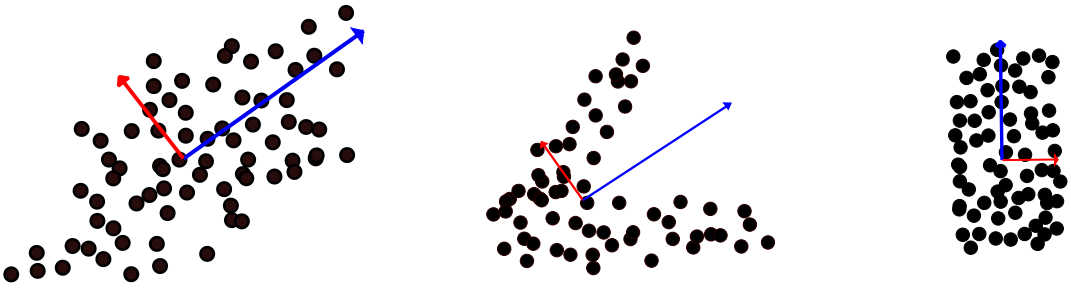

Figure 27. Illustration of using PCA for 2D cloud data.

The same principle works for 3D point cloud, from which an illustration can be seen in Figure 28. Since the points in segmented regions have already been filtered and they define a single aspect of the point cloud, PCA can be used efficiently for the data analysis. In this case, the main limitation for PCA comes from the accuracy of the segmentation and the sensor. The objects with more simple structure can be segmented more easily and

accurately. Objects with the more complicated design bring challenges for the segmentation algorithms to segment the object successfully repeatedly.

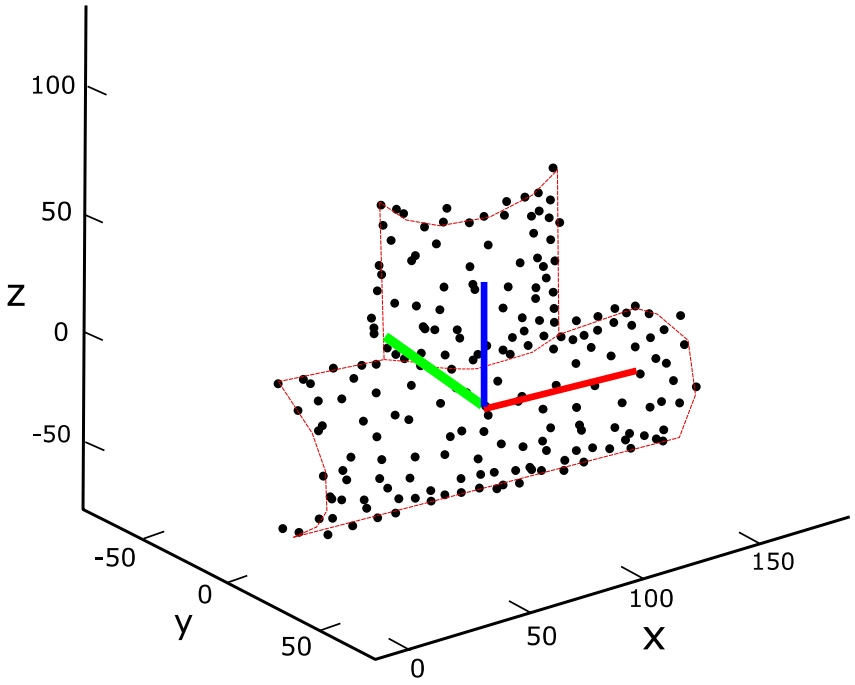


Figure 28. Three eigenvectors calculated for the segmented 3D point cloud of the T-pipe.

### 4.4.2 Localization routines

Localization routines can rely purely on the measurements or it can be done using the object's 3D-model as a reference for the measurements. If 3D-model is not accessible, localization features need to be taught for the system. For objects that can be segmented as one unique segment, PCA can be used to provide coordinate axis (see Figure 29). If the object has multiple segments, the localization can be done by selecting unique segments to define the coordinate system axis. For instance, one of the normals provided by the PCA for the segment will define the X-axis for the coordinate system, another segment the Y-axis after which the Z-axis can be calculated with cross-product. This kind of localization method will be very case-specific.

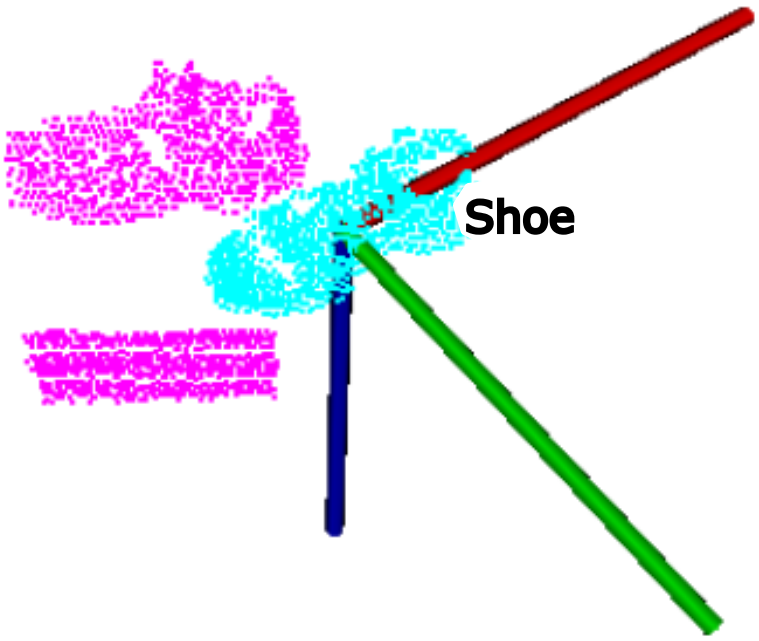


Figure 29. Example of the object recognition.

However if a 3D-model is accessible, it can be used for localization purposes. In that case, the localization will be about using both the reference information and the measurements to localize the object in the robot's workspace. Main limits for the use of 3D-models in localization come from the geometric properties of the object. It can be quite hard to use complex features like dual-curved surfaces to localize the object accurately.

# 5 ROBOT SKILL PROGRAMMING

Robot skills can be modelled as re-usable robot operations, which integrate and synchronize robot actions and sensor data in a consistent way (Heikkilä & Ahola, 2018). For example, a pick-up skill is composed of smaller tasks that the robot will perform in sequence with utilizing the sensor data. In this chapter, robot skills will be explained to the reader in the programming level and some examples of different skill sets will be shown.

## 5.1 Structure of the skill-based program

Easy and quick re-programming of robots by non-experts has been the long-term objective of the robot skill development. The use of robot skills has been proposed as answer to the high demand of flexibility in industrial production. Compared to the traditional robot programming, task-level programming constitutes a higher abstraction level. Pedersen et al (2016) explains the approach of task-level programming by dividing it into three layers: primitive, skill and task layers as can be seen in Figure 30. The task layer can be seen as the assignment that robot is performing. Tasks can then be divided to skills, which are composed of series of primitives. (Pedersen et al 2016.)

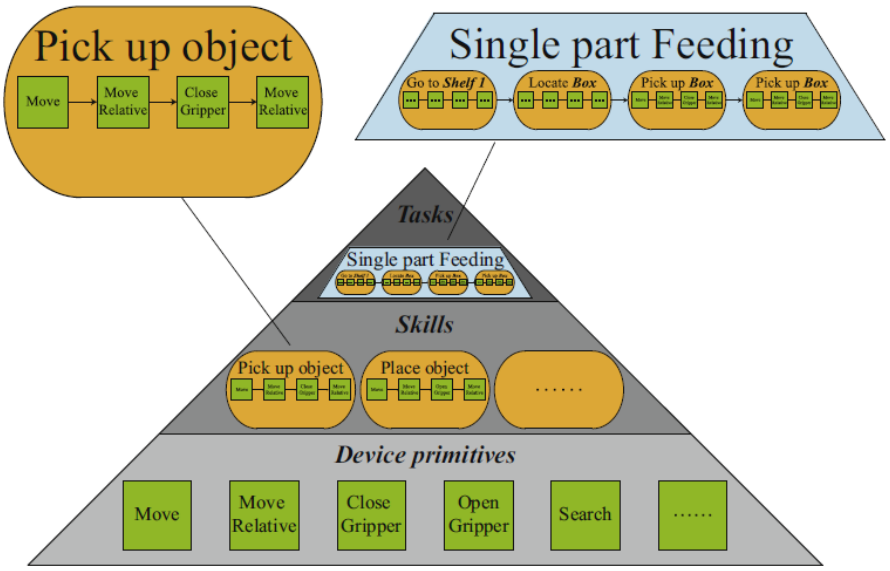


Figure 30. The three layers of tasks, skills and primitives with example of single part feeding process. (Pedersen et al 2016)

Bøgh et al (2012) noted that one of main reasons of using the skill-based programming is their object-centeredness. Classic robot programs usually consists of 3D-coordinates and actions being completed in the right place and time. “Pick up object” function requires 3D-coordinates of location to robot to move in the right position and then grasp the object.

### 5.1.1 Primitives

Primitives are the lowest level of programming that consists of what could be considered as the robot work sequence. In skill-based programming, the primitive layer is hidden from the operators and will be only exposed to expert users. The programmer will program the primitives needed for each skill. The skills are parametrized so that each working object will have its own primitives. For instance, pick-up skill will have very different primitives, whether the work object is a coffee cup or a large box. In addition, movement targets depend the location and orientation of the object. Depending on the work object, there might be need for use different gripper that the robot needs to comprehend and change automatically. In addition, approaches for grabbing can be vastly dissimilar. Primitives include the use of all the sensors integrated to the process that are needed for the specific skill. Examples of primitives can be seen in Figure 31. (Pedersen et al 2016.)


Figure 31. Example of different primitives in skill-based programming.

### 5.1.2 Skills

According to Pedersen et al (2016) each skill should be able to do following: perform the same operation regardless of the given input, able to estimate if the skill can be executed given the state of the world and the given input. After execution, it should be able to verify the result. Skill called “locate-object” would use some kind of camera or scanner first to localize the work object. For example, once the desired object is recognized and located, robot would be able to execute the skill “pick-up ball”. The pick-up skill would be composed of unique primitives that would be defined by the object and its location. This normal pick-up skill could be developed further so, that the robot will use force sensor to

sense the contact before grasping. That would then be a new skill “pick-up ball with contact force”. Example skills can be seen in Figure 32.


Figure 32. Examples of different skills in the skill-based programming.

### 5.1.3 Tasks

Both the task and skill layer will be visible to the end user, but only the task should be programmable by operators. This way the operator can use the robot to perform actions without the need of deeper level of programming. The task should give all the necessary parameters to the skill layer. Those parameters should include information such as the identification of the work object and the actions that needs to be performed to it. The objective is that the programmer will predefine and parametrize the skills and the end user will program the tasks in the factory. Some possible tasks are represented in Figure 33.


Figure 33. Examples of tasks in the skill-based programming.

## 5.2 Sensors for robots

Skills rely heavily on different sensors that will give the robot the ability to sense the real world in different ways. The sensors can be divided into different categories that represent different senses that enable the use of specific skills. Machine vision skills utilize 2D profile scanners and 3D depth sensors that will provide the robot an image about its surroundings. These sensors can be used for the robot to detect, recognize and localize the objects. Object detection requires that the state of the environment can be observed and any changes noticed. Recognition and localization skills require detecting unique features that can be reliable compared to the reference database.

Another type of typical sensor integrated to the robot is a force-torque sensor. Force-sensors can be utilized for the robots in many ways. With force sensor integrated into robot's flange, force-controlled motions are possible to perform. For instance, it can give the robot an ability to "feel the object" before grasping or follow the surface while keeping the contact. Force-control is very useful tool for overcoming all the localization errors that come from the sensors and software in the localization procedures.

Many safety features can be combined with machine vision based sensors and the force-sensors of the robot. For example, 3D-camera monitoring the robot's workspace can recognize humans entering into its field of view and stop the robot. Force-control can be used to monitor possible collisions between the robot and other objects. Too high contact forces can be programmed to activate the triggers that will stop the robot.

## 5.3 Different skill sets

A robot differentiates from basic CNC machines with compound of layers that need to collaborate properly. Normal CNC program is composed of coordinates that the machine runs in sequence. With robots, more intelligence can be implemented into process. All the sensors and the control systems will have to communicate efficiently together.

The robot system includes all the sensors, actuators and control systems integrated to it and each of them provides a layer to the program. Sensors will provide information about the real world that the control system can use in real-time for instance correcting the movements of the robot, activity of the tool or both. In Figure 34 can be seen the layers of an example robot skill.

| Force-sensor | Controller | Program | Robot | Gripper |
|---|---|---|---|---|
| Force values | Control-motion<br>- movement<br>- velocity<br>- acceleration | Start/Stop<br>Work sequence<br>Parameters | Poses<br>Motions | Open<br>Close |

Figure 34. Different layers in a robot skill program.

In order to use skill-based programming, most programs need to be fully reworked. As the whole concept of skill requires right kind of parametrization, most programs will not

be usable at their present form. Skill templates would need to be created from the primitives, after which the skills could be combined to form the tasks. There can be lot of layers in one skill as every sensor, controller, software and tool will have their own layer. In addition to performing the assigned skill, the robot should be able to determine the outcome of its actions. For instance if the task was to move the objects from the location A to the location B, the robot should be able to confirm that the object has ended in the location B.

# 6 GRINDING WITH LOCALIZATION

In this chapter, structure of the task in which the object is localized and grinded, will be presented. The task has been formed from smaller lower-level skills (see Figure 35). RoboDK will used to gather the reference information from the 3D-models of the work object as well as programming all the subprograms for the robot.

The process will start with global localization of the object, which gives an estimate of object's location and orientation. Accuracy of this kind of localization is not accurate enough to perform machining on the object, but it can be used to get the precise location of the object. Estimated position will be used to update the object's location in the simulation, after which the object's scanning movements can be programmed. In scanning skill, the robot will perform movements with 2D-profile scanner to get data from the object. After the data has been obtained, precise localization -skill can be used that will output precise location of the object. After this, the location is known with enough accuracy, that grinding skills can be applied. Grinding skills include straight seam grinding and level surface grinding skills. More detailed flowchart of the whole task can be seen in Appendix 1. The structure of this chapter will go as follows: at first, all the equipment used in the experiments will be introduced after which all the used skills will be explained in detail.

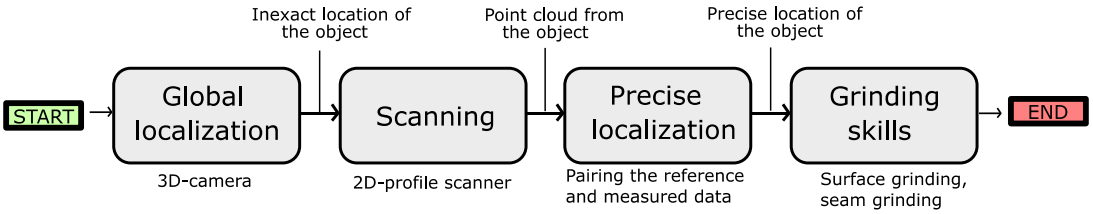


Figure 35. Lower-level skills of the grinding with localization -skill.

## 6.1 Equipment for the skill implementation

Grinding with localization -task requires an industrial robot with integrated force-sensor, 2D-profile scanner, 3D-depth sensor and grinding tools for the robot. Tools for the grinding as well as 2D-profile scanner are attached to the robots flange and only one can be equipped at the time.

### 6.1.1 Industrial robot

Industrial robot used in experimental tests is KUKA KR120 R2500 PRO (see Figure 36), which is a 6-DOF industrial robot. It has a payload of 120 kilograms and its reach is just short of 2.5 meters. It has footprint of 830mm x 830mm and it weighs approximately 1049 kilograms. Its precision is guaranteed to be at least ±0.06 mm. (KUKA 2018a.)


Figure 36. KUKA KR120 R2500 PRO. (KUKA 2018a)

KUKA robot controller (known as KRC) is the control system of KUKA robots. The newest version KR C4 that the KR120 2500 PRO uses, software architecture allows integrating robot control, PLC (Programmable Logic Computer) control, motion control and safety control all into one controller. The force-sensor and 2D-profile scanners attached to the robot's flange use PLC functions to transfer the data. (KUKA 2018c.)

KUKA system software (known as KSS) is the operating system of KUKA robot and thus one of the most important parts of its controller. It holds all the functions needed to operate robot system and it can be easily accessed by using KUKA smartPAD. The structure of KSS in the newest versions is based on Windows XP, which allows expanding the robot system with installing additional software. Up to 26 languages are supported for selection to the user interface. Basic functions of KSS enable the basic path programming from simple inline forms through to more expert programming with KUKA robot

language. User can also configure the I/O's for various field buses that includes for example Ethernet IP, Profibus, Profinet, EtherCAT etc. (KUKA 2018b.)

The KUKA Robot Language (known as KRL), is a proprietary programming language to control KUKA robots. All KUKA robots use KRL to create programs to perform. With KRL, robot movement paths are easily programmed and KRL enables user to use loop structures like LOOP, WHILE, IF, SWITCH and so on.

### 6.1.2 Force/Torque sensor

A force/torque (known as F/T) sensor system consists of the controller and the transducer. The system in the experiments uses ATI's Omega160 six-axis force/torque transducer and F/T controller. Transducer is located in the flange and it produces the force-torque signals. Controller is attached to fifth joint of the robot and it transforms the signals into values that can be sent to the robot (see Figure 37). The system can measure forces as high as 2500 N in the X and Y directions and 6250 N in the Z-direction (nominal to the flange) with the resolution being between 0.5 N and 0.75 N. The maximum measurable torque is 400 Nm in all directions with the resolution of 0.05 Nm. The accuracy is guaranteed to be ± 1 % at upper range values. (ATI 2010.)

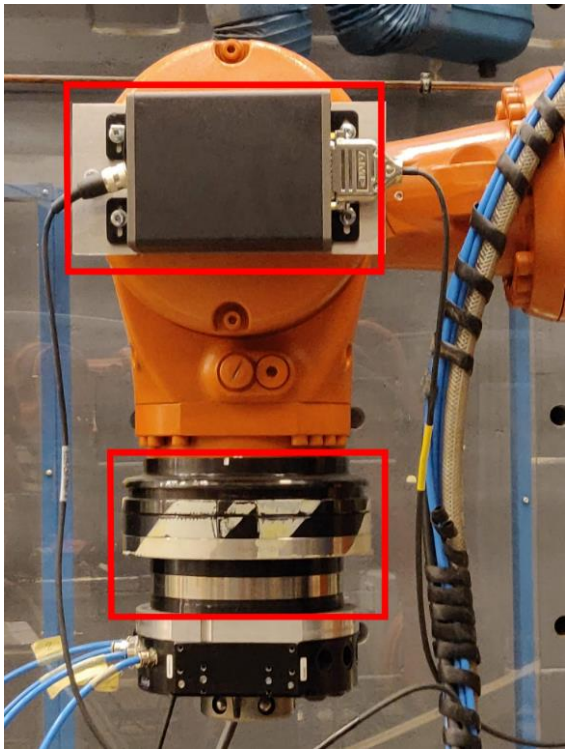

Figure 37. Controller (above) and transducer (below) of the force/torque sensor system.

### 6.1.3 2D-profile scanners

Micro-Epsilon scanControl 2700-50 is a 2D-profile scanner designed for the industrial use. It provides the accuracy needed for profile scanning and it can be applied with a protective cover for the harsh industrial environments. 2700-series has three different models; 2700-25, 2700-50, 2700-100, that are designed for different measurement ranges, as midrange ranges to 100, 200 or 400 mm. Image of the 2700-50 attached to the robot can be seen in Figure 38. (Micro-Epsilon 2019.)

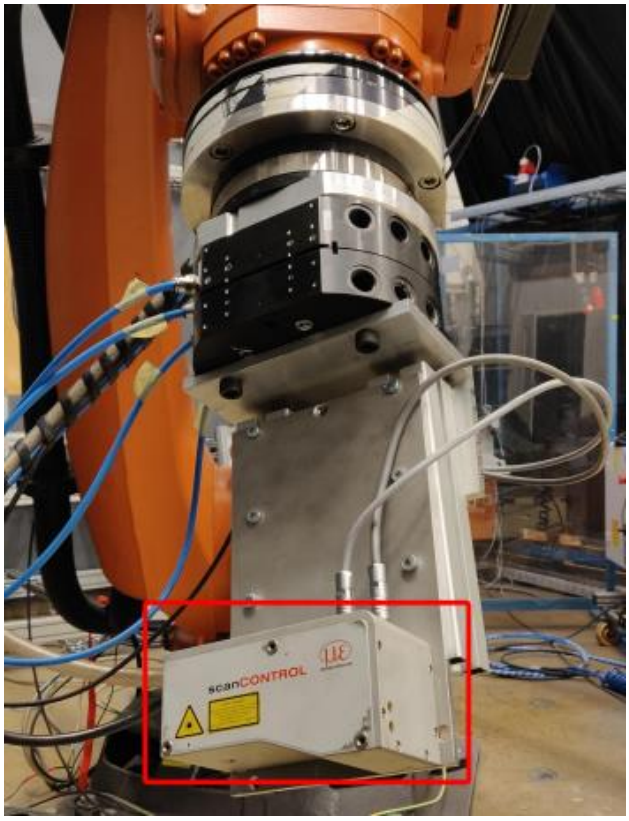


Figure 38. Micro-Epsilon scanControl 2700-50 attached to the robot.

Other 2D profile scanner used in this thesis experiments is Micro-Epsilon scanControl 2900-100/BL. It is part of the new series, which has replaced the older 2700-series. The 2900-series have for example smaller casing and higher profile frequency than the old 2700-series. There are four versions available for the 2900-series, which have different measurement distances as well as the reference resolution in z-axis. Image of the 2900-100/BL can be seen in Figure 39 and the detailed comparison between the sensors in Table 1. (Micro-Epsilon 2019.)

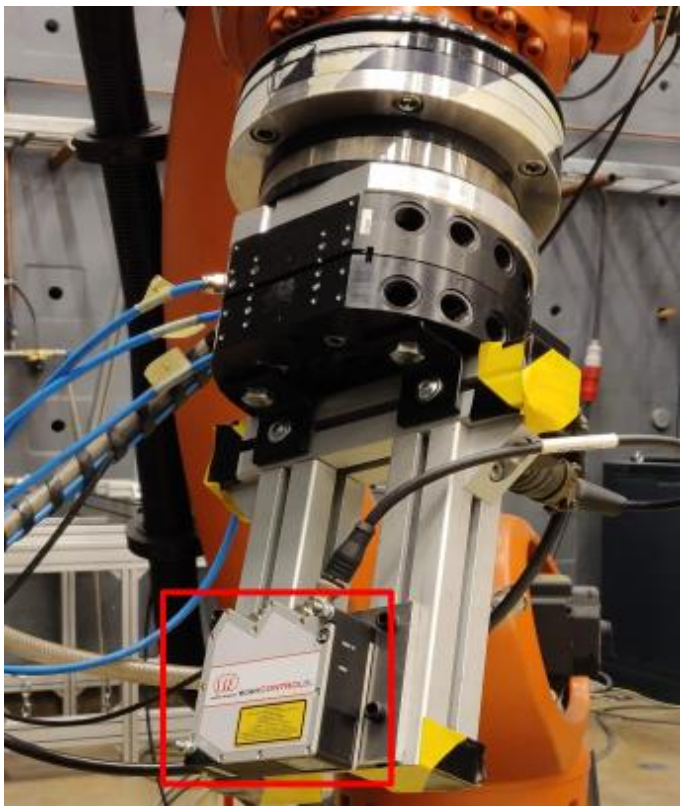

Figure 39. Micro-Epsilon scanControl 2900-100/BL attached to the robot.

Table 1. Characteristics comparison table of the between scanControl profile scanners. (Micro-Epsilon 2019)

| | **scanControl 2700-50** | **scanControl 2900 100/BL** |
|---|---|---|
| **Extended range in z-axis** | 100 mm | 265 mm |
| **Length of x-axis in midrange** | 50 mm | 100 mm |
| **Resolution, z-axis** | 10 µm | 12 µm |
| **Resolution, x-axis** | 640 points / profile | 1280 points / profile |
| **Profile frequency** | Up to 100 Hz | Up to 300 Hz |
| **Light source** | 658 nm (red) | 405 nm (blue) |

### 6.1.4 3D-depth sensor

The used 3D-depth sensor will be Intel's RealSense D415 (see Figure 40), which consists of pair of depth sensors, RGB (Red, Green, Blue) sensor and the infrared projector. The two depth sensors use stereo vision to calculate the depth and the infrared projector is

used to illuminate the objects with the structured light and thus enhancing the depth data. The image sensors provide up to 1280x720 resolution images, from which the depth can be calculated through disparity. Frame rate of the depth image can be up to 90 frames per second (FPS). The minimum range is set at 0.30 meters while the maximum range can be up to 10 meters but varies a lot depending on the conditions. Its field of view is 69.4 degrees horizontally and 42.5 degrees vertically. At minimum Intel guarantees the depth resolution of 1 % of the measured distance. (Intel 2019.)

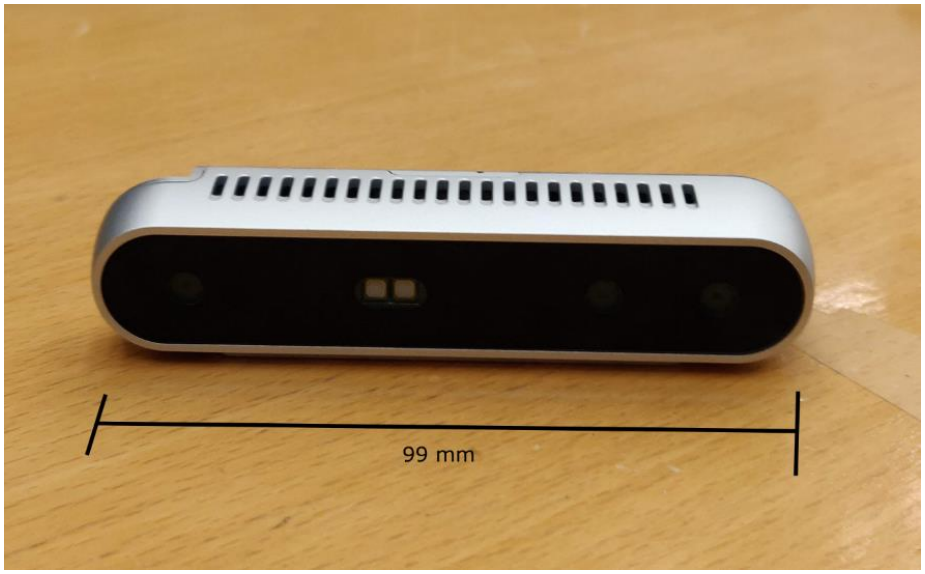


Figure 40. Intel RealSense D415.

### 6.1.5 Grinding tools

Two pneumatic grinding tools were made for the robot: angle grinder (Figure 41) and sander (Figure 42). Machines in the tools were manufactured by Atlas Copco and the mountings for the robot were done at the experiment lab. For both tools, initial values for TCP were first solved with KUKA's 4-point tool calibration method (explained in 2.5.2). Both TCP's were then corrected after trial runs, due to the difficulty of visualizing the same spot for the 4-point method. Tools were changed manually for the robot between different grindings.

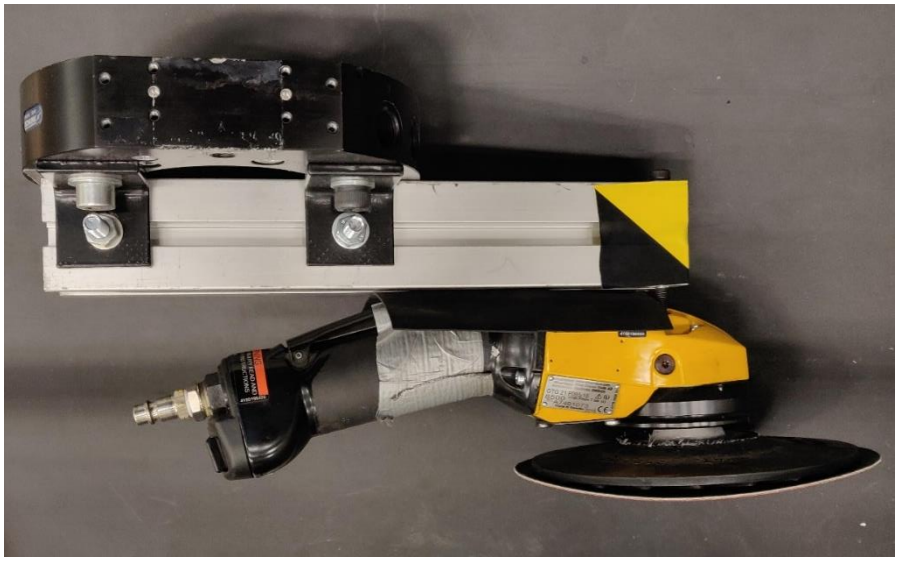

Figure 41. Atlas Copco GTG21 F085-18 angle grinder.

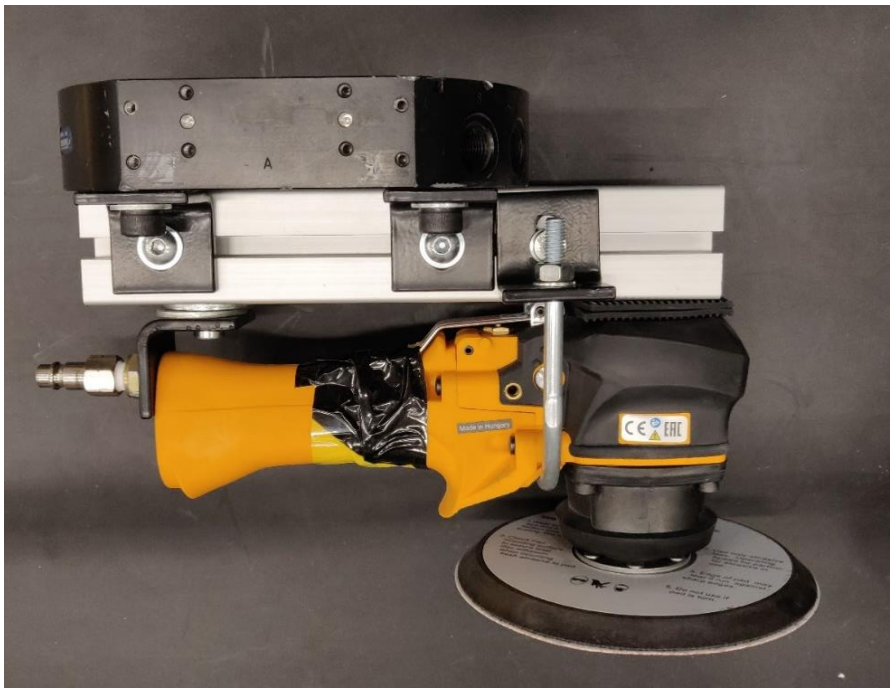

Figure 42. Atlas Copco LST30 H090-15 sander.

## 6.2 Object global localization

The first step of grinding with localization -task is to estimate location and orientation of the work object within the robots workspace. Main objective of the global localization is to provide the robot a coordinate system with such accuracy, which ensures that the scanning movements will be performed within enough proximity of the object.

As the robot's workspace can be multiple meters in diameter, single or multiple, external 3D-depth sensors are used. All the sensors will be calibrated relative to the robot, so that the robot's controller can transform the locations and orientations from the sensors coordinate system to robot's base coordinate system. 3D-depth sensors will be used due to their long measuring range and ability to capture the 3D-image at once, whereas scanning would require time-consuming movements to be performed. Drawback is that the localization will be less accurate. Due to this, it is more practical to first get the estimated location and then perform the scanning movements only for the desired sections. If the object can be placed in the known positions within enough accuracy, the global localization -skill could be passed. However, with global localization the object can be placed quite freely in the workspace. The accuracy of the location is not as essential as estimating the orientation of the object, since the following scanning movements are represented in the 3D-models coordinate system. Coordinate system of the 3D model is inside the object and the scanning moves being pointed towards to the object, the measuring area will be more sensitive to the orientation. If the orientation wobbles too much, 2D-profile scanners measurement "cone" will not be enough to capture the desired features of an object. Illustrative figure of the global localization setup can be seen in Figure 43.

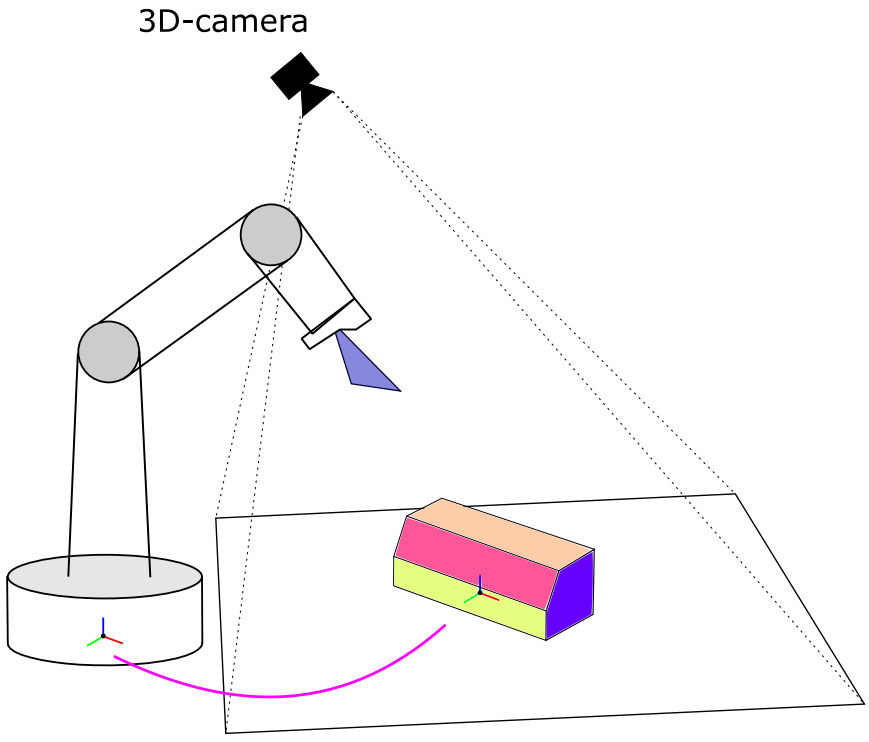


Figure 43. Setup of the global localization.

In practice, the sensor will provide 3D point cloud as an image of the area. In the best-case scenario, segmentation algorithms are able divide the point cloud into areas, which all represent different surfaces (different colors in Figure 43). For each segmented surface, surface normal and center of the plane will be calculated with PCA, which both are represented in the robot's coordinate system. Corresponding surface parameters will be gathered from the 3D-model of the object in RoboDK. The program will then calculate a coordinate system based on the three surface normals, which will describe the orientation of the object uniquely. Because the same surface normals will be gathered from RoboDK in 3D-model's coordinate system, the orientation of the coordinate system should be very similar. Location on the other hand, will be estimated by placing one reference point in the center of top surface and comparing it to the center point of the corresponding segment.

Gathering the surface parameters in RoboDK will be accomplished by manually adding points to corresponding surfaces in the 3D-model and calculating the surface normals from the added points. In Figure 44 is shown the example where three points have been added to three surfaces. Points will be added and the surface normals will be calculated from these points by Python programs running through RoboDK API.

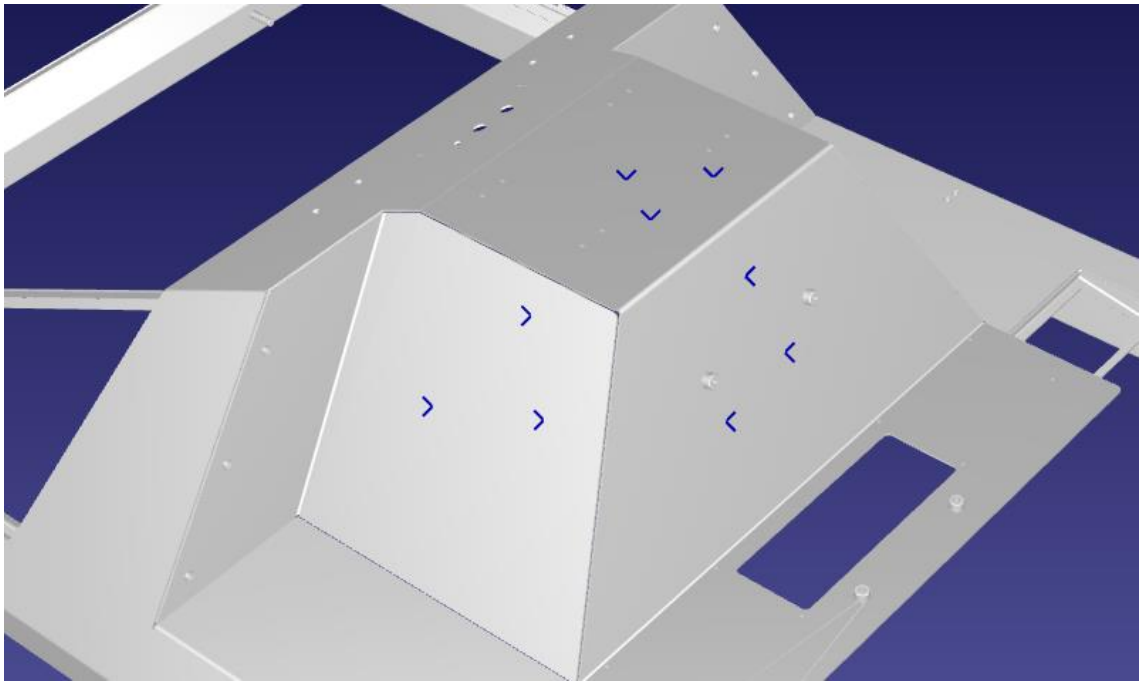

Figure 44. Extracting the surface parameters from the 3D model in RoboDK.

Three points per surface is required to calculate orientation of the normal unit vector. The starting point of the vector can be any point of the three points in the surface. The normal

vector is calculated by first calculating two vectors between the three points (for example from point 1 to point 3 and point 1 to point 2). Cross product of these two vectors will give the orientation of the normal vector. After the vector is calculated, it will have to be ensured that the vector is pointing outwards. Program makes a presumption that the 3D models origin will be in the center of the object, and uses the origin to create vector between origin and a point in surface. Dot product with these two vectors, one from the origin to the surface and the surface normal should be parallel. If not, the direction of the normal vector will be flipped.

After the surface normals are known both 3D-model coordinate system and the robot's workspace, location and orientation of the 3D-model's coordinate system in the robot's workspace can be calculated. The accuracy of this localization depends a lot on the sensor, even though it can be accurate enough for tasks, which are less sensitive to localization errors. The calculated coordinate system will be used to update the location of the object in RoboDK, after which the next skill can be performed.

## 6.3 Programming the scanning movements

After the objects location has been updated in RoboDK, scanning movements can be programmed for the robot. For most machining processes, position of the object is the not known accurately enough without scanning. Scanning of the object is conducted with 2D-profile scanner due to their high-level accuracy. The updated position from global localization should ensure that scanning movements are performed within the sensor measurement range. Usually, measuring range can vary up to 100mm from the optimum distance and still get satisfactory results. Next, the programming routine for the scanning movements is introduced.

As explained in the 4.4.1, for the localization of plane-featured objects it is required to obtain points from at least three divergent surfaces. Scanning movements are a compromise between the needed accuracy of the localization and cycle time of the sequence, as longer scanning movements are more time-consuming but will result better accuracy for the localization. Scanning movements will still need to be performed near the target surface or seam. Due to localization and manufacturing errors, the target locations in other parts of the object will vary too much for grinding.

With 3D-model of the object, programming the scanning movements for the robot is quite fast in RoboDK. However, it requires that not only the robot and work object have the actual 3D-models but also the scanner with the real TCP. Nowadays, most tools have accurate 3D-models available, but some of them can actually be too accurately modelled to use in RoboDK. One of the µEpsilon 2D-profile scanner used in the experiments in this thesis was so accurately modelled that RoboDK visualization slowed down significantly. For this case, there were no need for such accurate model of the tool so a simpler model of the scanner was modelled. The model was also added extra visualization of the scanner measurement range to make it easier for the user to picture the measurement range. Calibrated TCP from the actual robot was then updated to RoboDK, which allowed visually moving the 3D-model into right location. Updating the TCP does not move the geometrical body to its correct position, and it required moving it manually so that TCP matched the geometrical body.

Calibrated TCP is the scanner's origin of the measurements. It was shifted to middle of the measurement range. This will help to create the movement targets more easily. In Figure 45 can be seen the newly modelled version of the µEpsilon 2D-profile scanner. The blue section of the measurement cone is the optimum measurement range and TCP has been set in middle of the blue section.

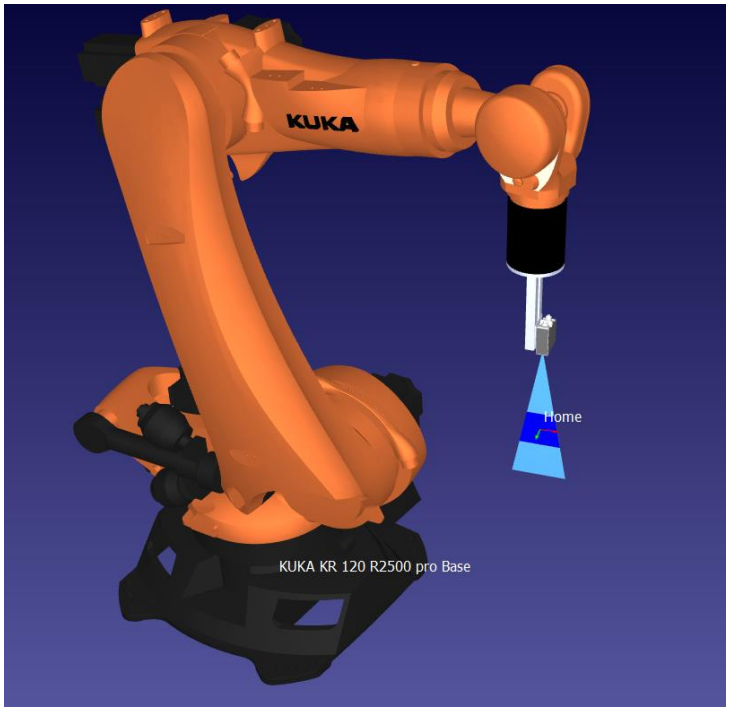


Figure 45. µEpsilon 2900-100BL 3D-model with measurement range visualization.

As the TCP has been set in middle of the optimum the distance, placing the targets can be done with RoboDK by its function called “Teach target(s) on Surface” that will place a target attached to the objects surface. Targets will only need to be shifted to the desired poses and it can be done by visually rotating the target. In Figure 46, three targets have been set so that the scanning movement will scan the two sides of the seam and the top surface.

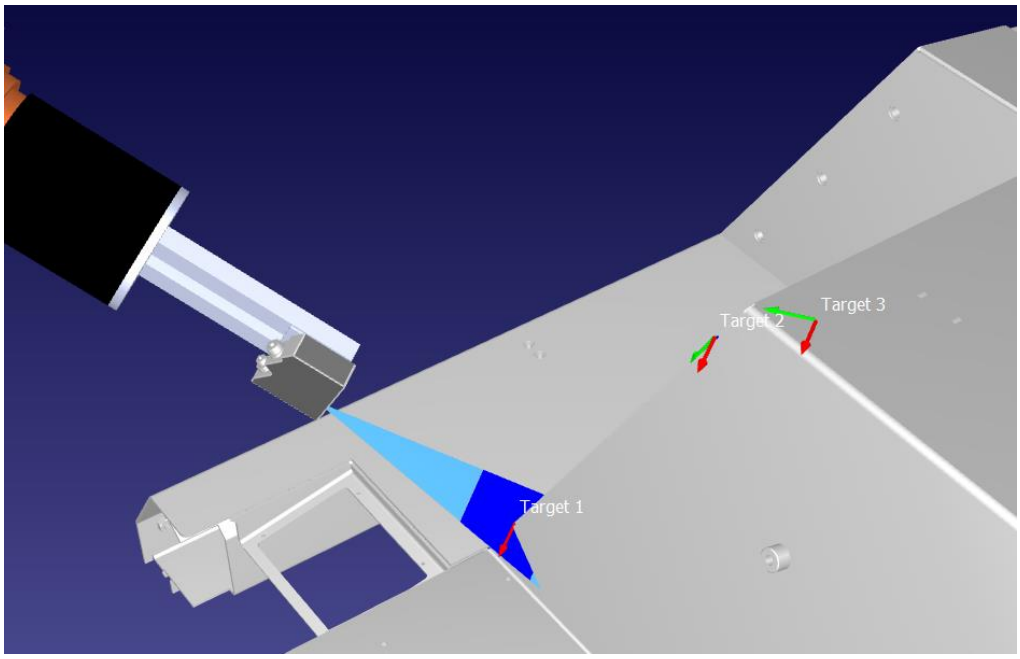


Figure 46. Targets for the scanning movement.

After targets have been positioned correctly, scanning movements can then be simulated in RoboDK. Simulation will give a notification if the movement cannot be completed due to the robot’s singularities or the lack of reachability. If the path cannot be performed, the object need to be moved for better area. Once the movement has been confirmed to be plausible and work as expected, a program can then be transformed to the robot’s own language.

The program in RoboDK is written with Python, which allows the same program to be used to perform the simulations and to be transformed into robot’s own language. The inputs for the program are the targets for movement. Because the TCP was shifted in to the middle of the measurement range for easier target generation, it needs to be shifted back to the origin of the scanner. The program will shift the TCP and at the same time shift the targets into same direction so that the movement will remain unchanged. In transformation to the desired robot language, all the necessary I/O’s that the scanner and

its software need, will be added automatically. The program can then be transferred straight to the robot through TCP/IP connection or manually with USB.

## 6.4 Precise localization

2D-profile scanner will provide a point cloud that is represented in the scanners own coordinate system, which location is known as TCP for the robot. However, for the robot to utilize the point cloud in localization, the controller needs to transform them into the robots base coordinate system. This will be done in real-time by adding together the known location of the flange and every points X, Y and Z values. This way the points will be represented into robot's base coordinate system. In Figure 47, the loop will be illustrated where the wanted expression is shown with purple line.

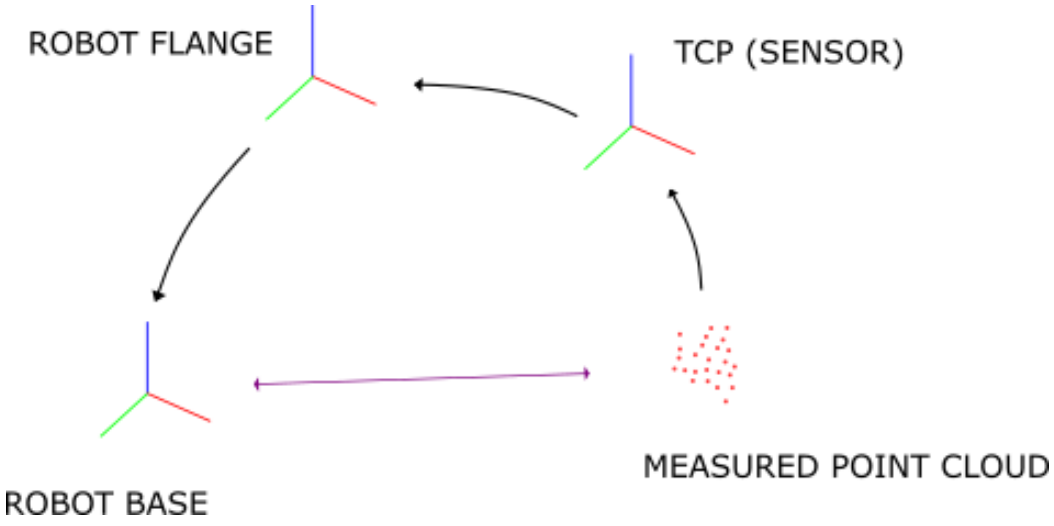


Figure 47. Transformation of the point cloud into robot base coordinate system.

Point Cloud Library (known as PCL) will be used to process the point cloud. PCL is an open-source library of algorithms for point cloud computing tasks. Algorithms are integrated into RoboScanner, which is the same software that is used to collect and visualize the measurement data. After scanning has been completed, filtering algorithms are used to get rid of outliers and smoothening the noise-level. After the data has been filtered, the segmentation can be performed.

After the point cloud has been filtered and segmentation has been performed, at least three divergent surfaces should be successfully segmented. These three segments will provide the parameters that can be paired with reference information. The reference parameters will be retrieved with the same method as for the global localization. However, instead

of computing the position in RoboDK, the reference parameters will be sent to pose estimator in RoboScanner. Corresponding information could be gained manually with user connecting the segmented surfaces with the same surfaces from the 3D-model. However, for this case RoboDK will provide the same information.

Three normal vectors from three diverging surfaces is enough for the pose estimator. The pose estimator is integrated into RoboScanner. RoboDK will sent the reference surface parameters for each surface (X, Y, Z, RX, RY and RZ) to Roboscanner through a socket. Pose estimator's simplified operation principle is iteratively to fix the position parameters so that the distance between the measured point and equivalent surface in the 3D-model will be minimized. After the position has been estimated, the position of the object will be sent back through same socket and used to update the object's location in RoboDK. If the same surfaces are scanned, the reference information does not need to be changed. Figure 48 will illustrate the principle, where the segmented point cloud is fitted to the object's 3D-model. Three colors represent different segments and red points are the dismissed points, which are located in the seams.

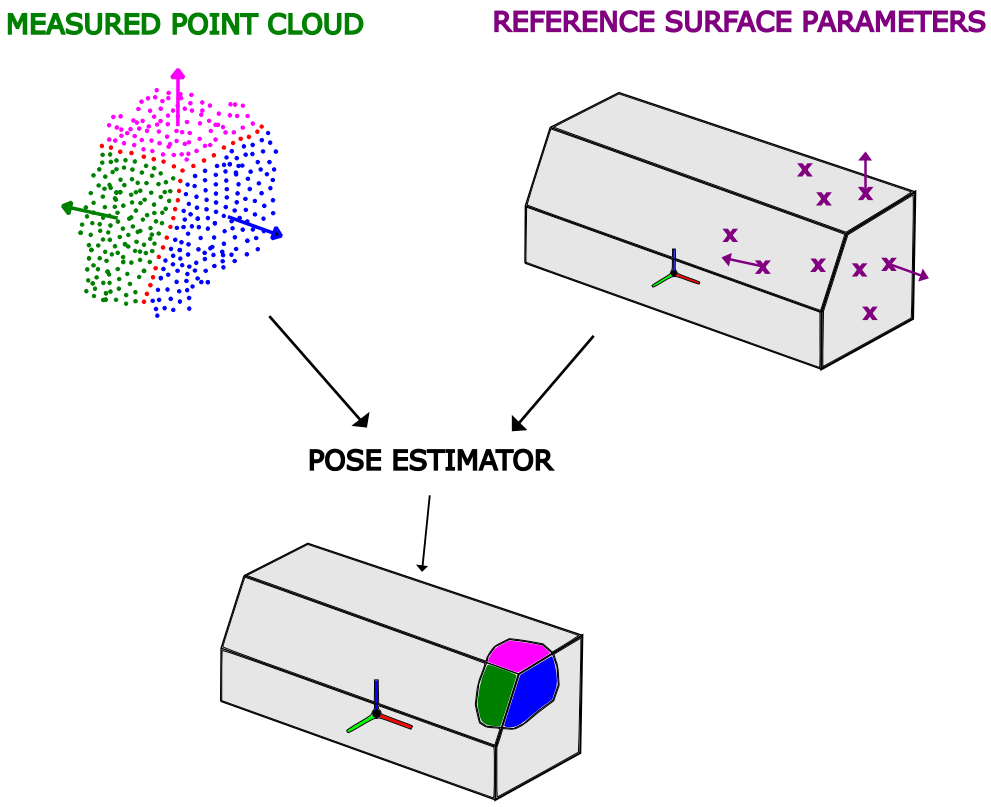


Figure 48. Pairing the measured and reference surface parameters together.

## 6.5 Programming the grinding skills

After the object's coordinate system has been updated in RoboDK to match the real world location, grinding programs can be programmed. Grinding operations are composed of different lower-level skills, which are programs in the RoboDK written with Python. These grinding skills will only require four inputs to determine the whole program. The four inputs are target poses in the object's frame and they describe the location and orientation of the area that needs to be grinded.

Even after precise localization, force-controlled motions will be enabled for the robot. This means, that the robot will use the integrated force sensor to sense the contact with the object and apply the appointed nominal force against the surface. The main benefit of the force-controlled motions is to counter any manufacturing defects and localization errors. However, due to very nature of the grinding, it is also beneficial to apply nominal force to make the grinding more effective. In other processes, it can be used only to sense the contact without applying only minimal amount of nominal force.

Two lower-level grinding skills have been created: straight seam rounding and level surface grinding. Seam grinding will be performed in two stages: using the angle grinder to remove the excessive material from the seam and round the seam with the sander. The sander will only be used to smooth any angularity that will be left from the angle grinder. Surface grinding will use the sander to grind the area that is limited with the four given positions. With these two skills, most objects can be fully grinded with the obvious limitations given with the robot and its tool's physical dimensions. RoboDK will simulate the program and show the movement path, so it can be easily analyzed. It will also notify automatically if the object is currently out of reach or the moves would cross singularities. In addition, if the environment is accurately modelled, RoboDK can be used to check any collisions with the environment. Once the program have been simulated and proved plausible, a program can be translated in the robot's programming language. Selected grinding program will be transferred same way as the scanning program, either using TCP/IP connection or with USB. Next, the structure of these grinding skills will be introduced in more detail.

### 6.5.1 Straight seam rounding

Location and orientation of the seam will be described with four poses, pair of poses in the beginning of the seam and another pair in the end of the seam. The pair will include poses from the both surfaces in close proximity (see Figure 49). When they are set in the object's frame, target locations will be given in the object's coordinate system. The same function of RoboDK called "Teach targets on a surface" will be used to make sure that the XY-plane will be placed on a surface, which means that the Z-axis will represent the surface normal pointing towards object. Targets will only need adjusting orientation around the z-axis and fine-tuning the location.

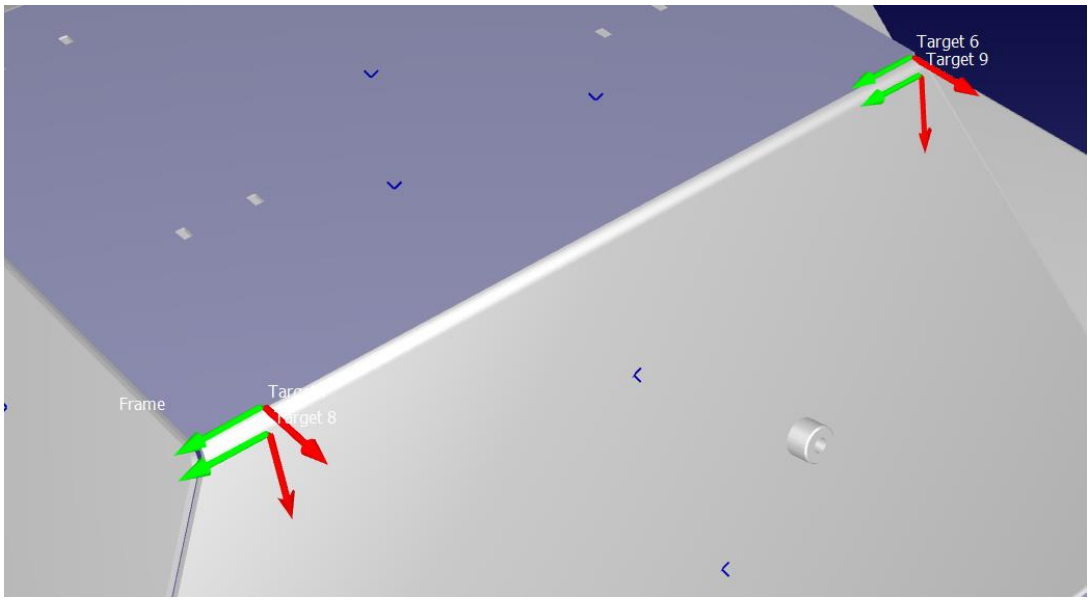


Figure 49. The four targets needed for seam rounding.

As the four targets will present the seams location and orientation, rest of the program can be created based on these four targets. As these four targets are not enough to round the seam, some extra targets that are located in the middle of the seam are needed. All the extra targets could be placed in the RoboDK manually, but it would be too inaccurate and time-consuming. Because grindings will be performed with force-controlled motions, the extra targets do not need to be exactly on surface of the real object. It usually is not even possible due to manufacturing defects that will make the manufactured object deviate from 3D-model. This will be countered by taking the contact position for example 1 centimeter away from the surface, after which the robot will sense the contact point by slowly moving towards the surface. In Figure 50 can be seen the example, where three extra targets are visualized. The extra targets are located close to surface in the simulator

and their orientation will change gradually from the surface to surface. These extra targets can then be used in the vertical and rounding grinding movements.

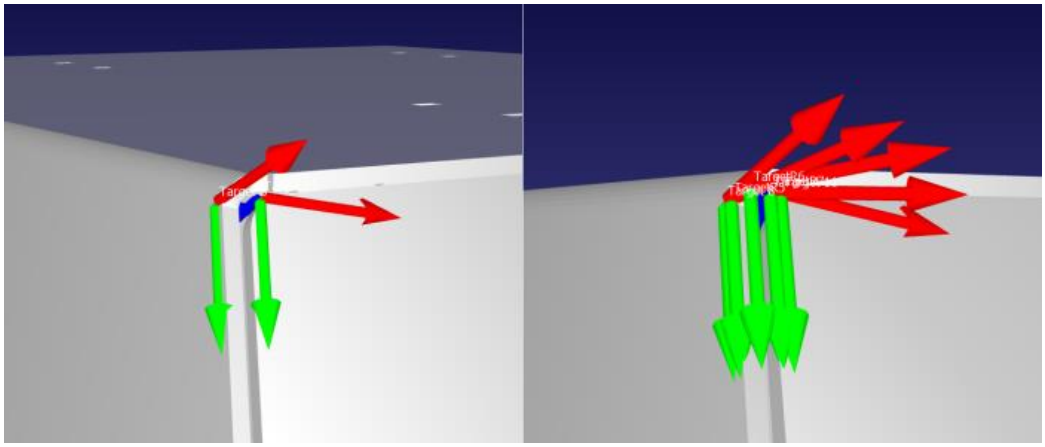

Figure 50. Visualization of the three extra targets created by the programs.

The calculation of extra targets will start by calculating the closest point between the two vectors (z-axes) and using that as origin. Once the origin has been calculated, the algorithm will calculate desired number of targets to the surface with equal angles in 3D space. Because the extra target will form a right-angled triangle with the origin and the other given pose, the distance between origin and target location can be calculated. Once the angle and the distance has been calculated, they can be used to create a vector that starts from the origin. This will give the target's location in 3D space. Orientation of the target can be solved by rotating pose of the original target gradually around the Y-axis. Illustrative image of this calculation can be seen in Figure 51, where three mid-positions has been calculated for the seam.

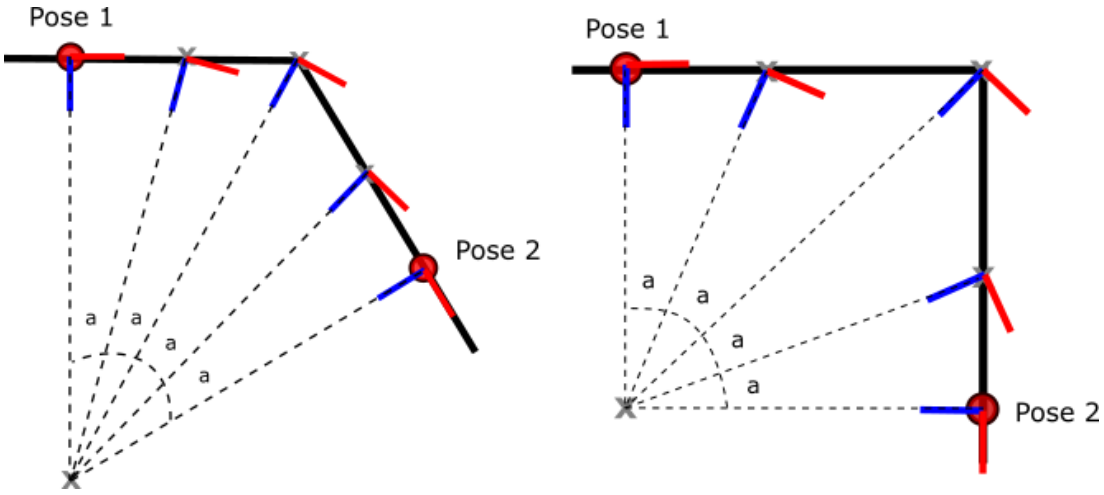


Figure 51. Illustrative figure for calculating the rounding target.

After extra targets have been calculated for both the start and the end of the seam, they can be used for vertical grinding movements that will be performed with the angle grinder. Due to the grinding angle, movements will be done “backwards” to ensure smoother movement. Nominal force for the grinding was varied from 10 N to 20 N, to test different grinding cycles. The full cycle of the program in RoboDK and an image from the simulation can be seen in Figure 52.

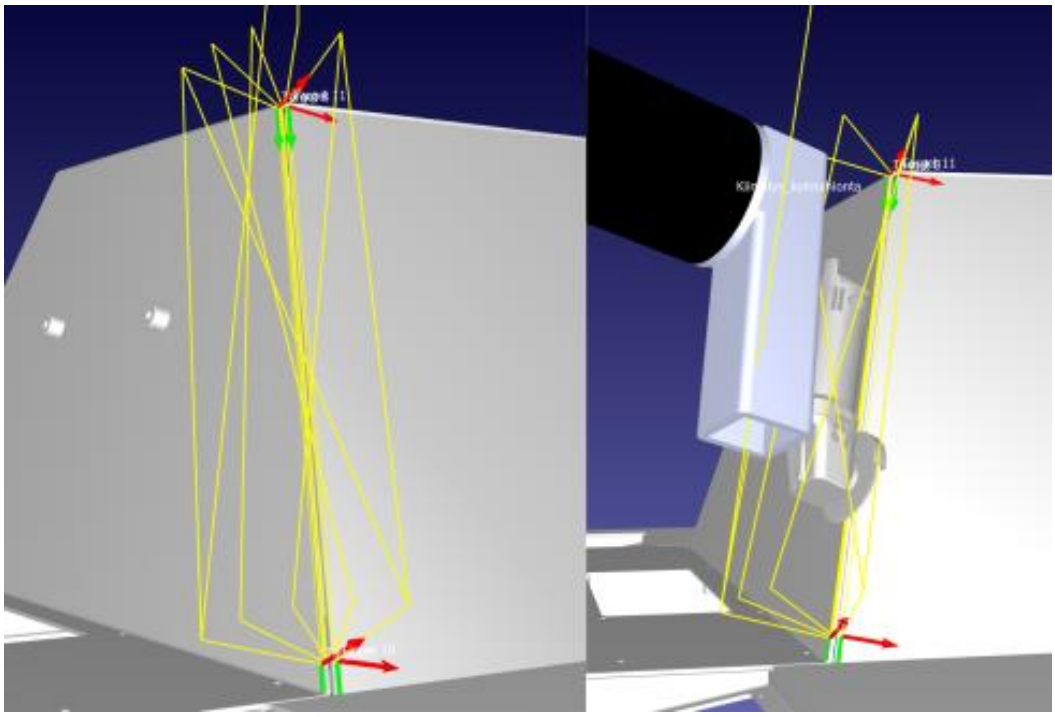

Figure 52. On the left simulated full vertical grinding cycle path and on the right in the middle of the simulation.

After the angle grinder has removed excessive material from the seam, the shape of the seam should be quite pointed and the purpose of rounding movements is to smooth the seam. For that, tool will be switched to the sander. For the rounding, the starting point and the ending point will be both shifted half tool’s diameter inwards. This is due to fact that TCP of the sander will be in the middle of its flange. Rounding process is designed so, that it will start from the other end of the seam, perform rounding movements from surface to surface and move linearly along the seam. This will then be repeated for the whole length of the seam. The length will be calculated from the original four targets and interval for the rounding will be determined by a parameter. This way the number of rounding movements in a seam can be determined by the programmer. Other main parameters for the rounding are grinding speed, tool diameter and nominal force applied

to the grinding process. The modifiability of the program is seen substantial, since all the parameters have direct influence on the quality and adaptability of the skill.

Two different of programs have been developed for rounding. The first one will use same targets as the vertical movements but instead moving in end-to-end direction, the movement will be performed linearly from surface to surface. Even though the targets are not in an arc that would describe radius of the rounding, force-control will maintain the tool in contact with the surface. In Figure 53 can be seen the rounding path with the first method. The targets used are the same as pictured in Figure 50.

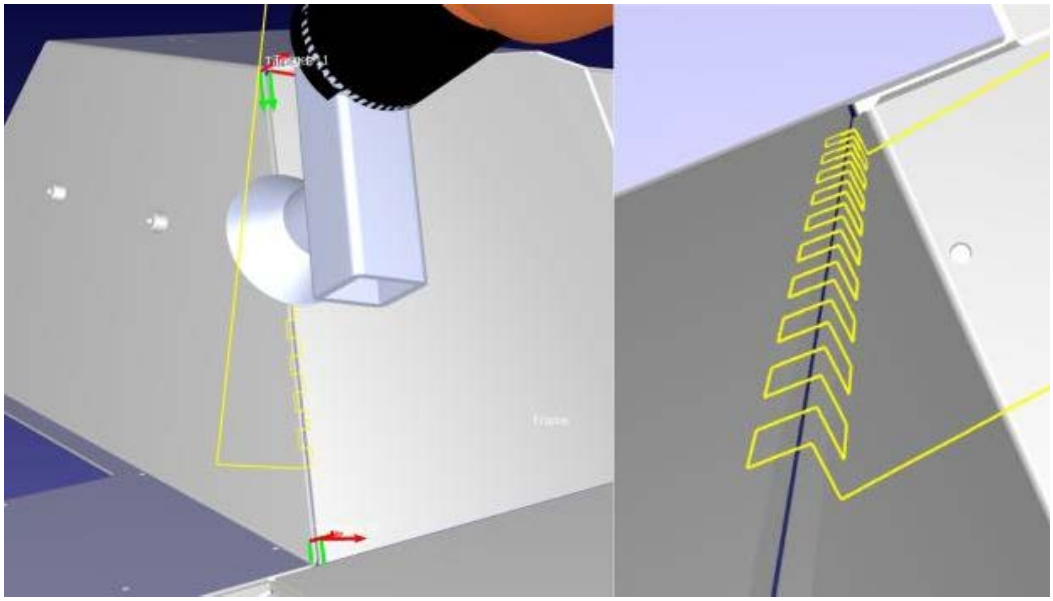

Figure 53. Linear movement based rounding path.

The other method for rounding movements is based on circular movement. Circular movements are the third type of basic movement used in robotics, in addition to linear and joint moves. Circular movement requires two inputs, mid-target that is in the arc and ending point of the move. Mid-target in the arc needs to be calculated and ending point is the target on the opposite surface. The target for rounding will be calculated in a similar way as it begins by calculating the same closest point between the Z-axes. Then instead of calculating the target location on the surface by forming the right-angled triangle, it will calculate the radius from the distance between the original targets relative to the origin. This will be then used as radius for the arc and the target will be located in middle of the arc. The rounding target will be slightly inside the object, but the force-control will once more correct the movements in that direction. The calculation of the target for

circular move is illustrated in Figure 54 and the circular movement -based rounding path can be seen in Figure 55.

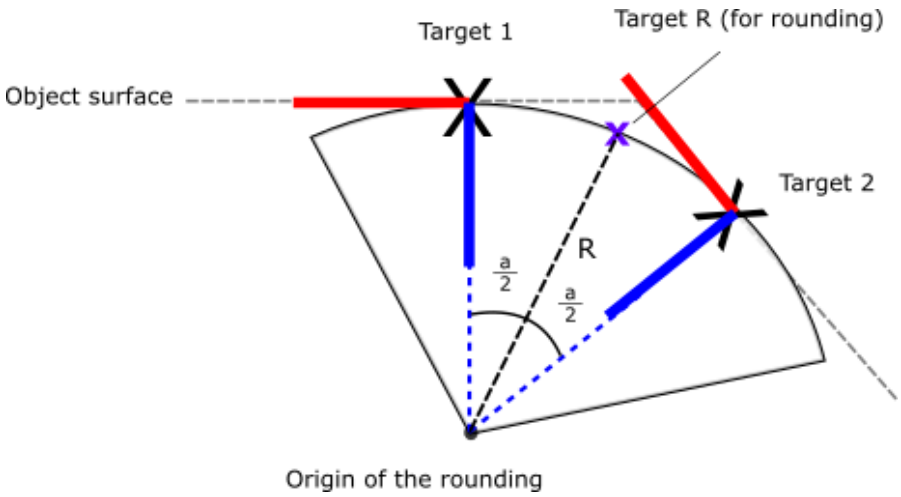


Figure 54. Calculation of the target for the circular move.

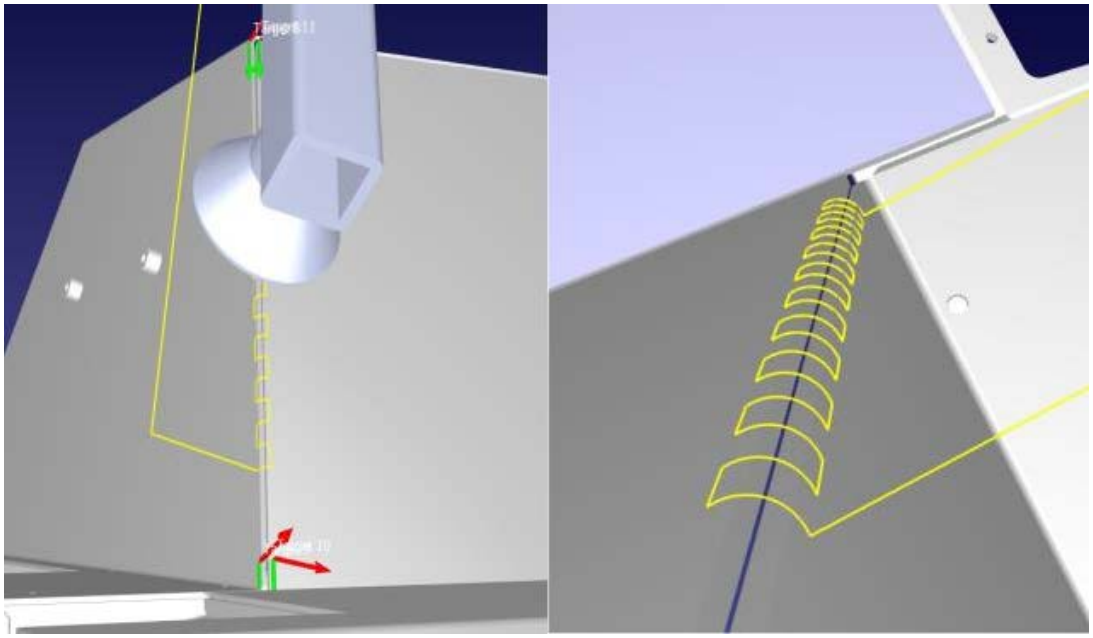

Figure 55. Circular movement based rounding path.

Vertical and rounding grindings, based either on circular or linear movements, will be performed for each target seam. New program for other seam can be created by simply placing new targets, as all the approach moves will be also created based on the original targets. An example of the complete movement and grinding path for the single seam can be seen in Figure 56.

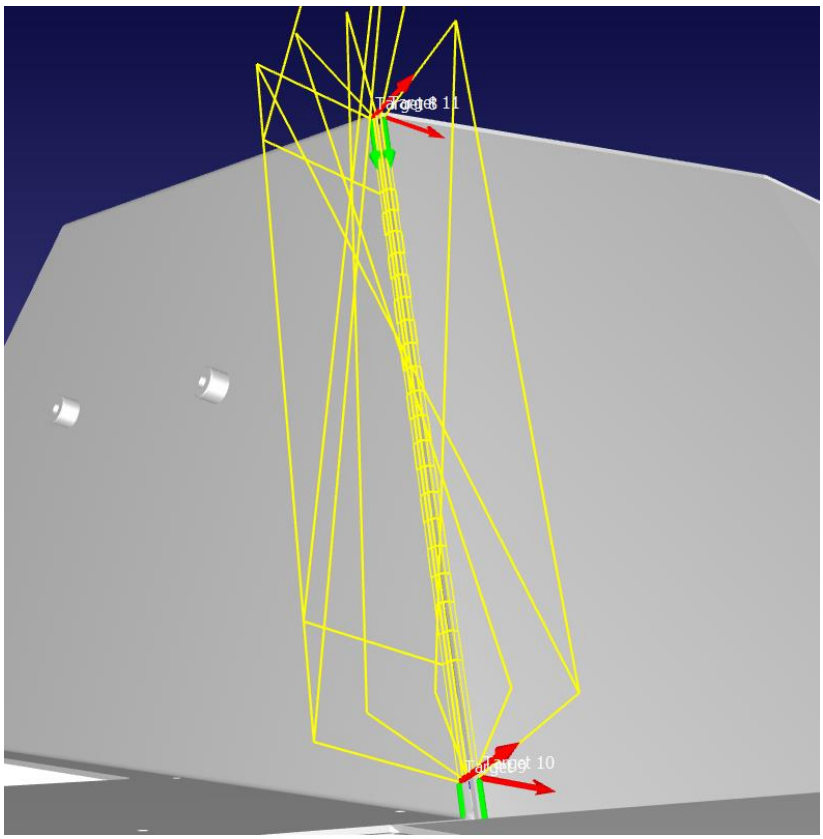


Figure 56. Combined vertical and rounding seam grinding paths for a seam.

### 6.5.2 Level surface grinding

Level surface grinding will need the same number of inputs as the seam rounding: four targets to define the area that need to be grinded. The targets will be taught similarly to the seam rounding but in this case, they will all be located on the same surface. The skill will have the same modifiable parameters such as grinding speed, tool diameter and applied nominal force for grinding. However, instead of interval, the parameter will be overlap of grinding between different crossovers. Tool diameter will be used to calculate the new targets that are inside of an area, so that edge of the tool will not cross the target area. Even if the defined area is not a square or even symmetric at all, the program will interpolate the targets using the same parameters. Overlap parameter allows modifying the overlap of different intervals to optimize the grinding. Interval length for the program can be calculated based on overlap and tool diameter parameters. In Figure 57 can be seen an example of the tool path with given four poses.

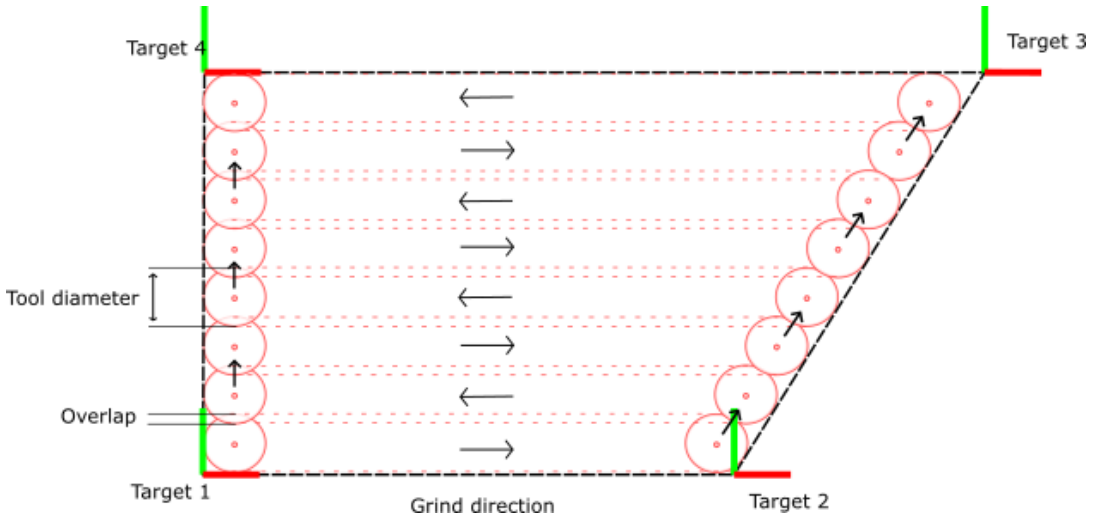


Figure 57. Principle of the path planning for the surface grinding.

As said earlier, the program will start with calculating the new poses that are inside of grinding area. For that, the program will calculate vectors between the neighboring input targets. Calculation is illustrated in Figure 58 where the two vectors are represented by yellow and blue colors. Radius of the tool is also known and shown with purple color. A unit vector pointing towards the desired point P (red) will be calculated by adding the two side-vectors together and normalizing it. Length for the desired vector will be solved from the right-angled triangle, once the angle between the two vectors (alpha) has been calculated and divided by two. After direction and length of the vector have been solved, location of the P can be solved and used as the location for a new target. This will be done for all four input targets, after which they all will be located inside the defined area.

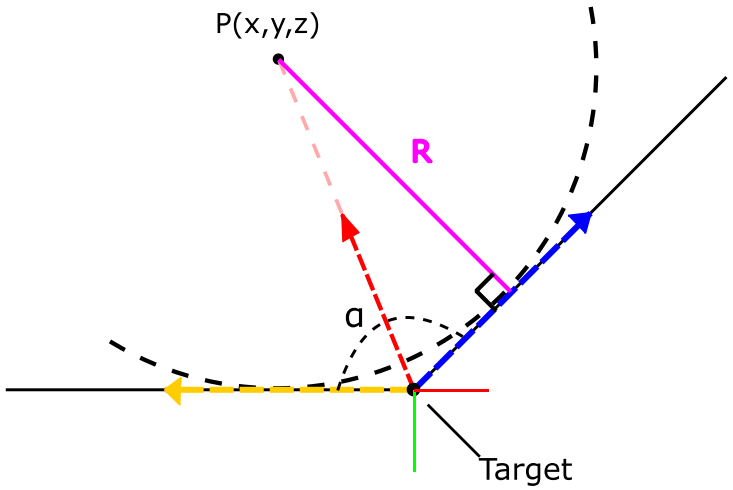


Figure 58. Calculation of the target location inside the area.

Grind direction is set by the first two input poses. Number of crossovers required for the area will be calculated by dividing the perpendicular length of the area with value of the interval. The interval represents the length between the parallel crossover paths. Because the result will not be an integer, it will rounded up and used to calculate the new interval value. This new smaller interval value will mean that the overlap will be slightly bigger than the original set value. However, for grinding movements, it is more preferable to grind with more overlapping than leave gaps that have not been grinded. In cases where the defined area is not a square, the program will calculate length of the shift based on the angle and interval value. This way the paths will cross each other evenly in the whole area. Example of the programmed paths that has been simulated in RoboDK can be seen in Figure 59 and Figure 60.

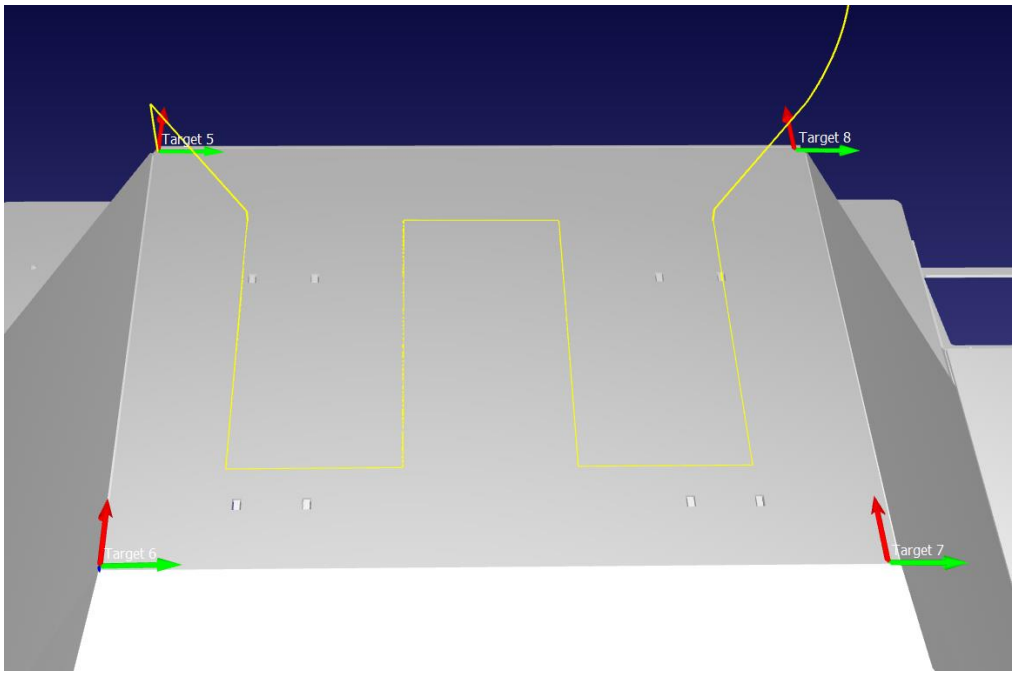


Figure 59. Surface-grinding path for the area defined by the four poses.

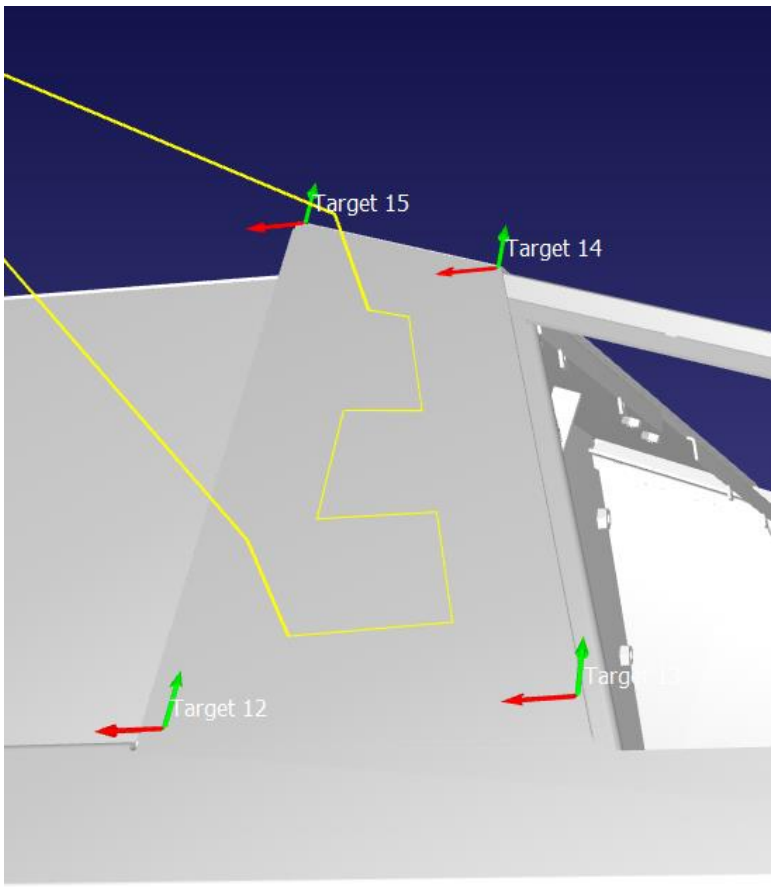


Figure 60. Surface grinding being simulated in RoboDK.

# 7 RESULTS AND DISCUSSION

In this chapter, results of the implementations will be shown as well as analyzed based on the multiple experiments. First, raw 3D image and segmented point cloud captured from the 3D-depth sensor will be shown and analyzed without explaining all the numerous parameter settings. After which, scanning movements that result the point cloud will be represented and fitted into the 3D-model. Finally, the seam before and after grindings will be represented. Both grinding skills were tested but actual grinding results were emphasized on the seam grinding, due to much more complicated nature of the rounding process.

## 7.1 Global localization

In global localization, 3D-depth sensor was installed in the robot cell so that it was far enough to capture the objects largest surfaces but close enough that the point cloud was as accurate as possible. After which, the sensor was calibrated relative to the robot. RealSense D415 provided the colored 3D-image (see Figure 61), which was used as illustration for what the camera actually sees.

Figure 61. Colored 3D-depth image provided by the RealSense D415.

Next, the point cloud needed to be segmented to use for localization purposes. The segmentation algorithm divided the point cloud into planes, from which the three largest non-parallel surfaces was selected. With the right setup for the segmentation parameters,

three largest surfaces were segmented as their own segments (see Figure 62). For each surface, PCA was used to calculate the coordinate system where the Z-axis (blue) represented the surface normal. These surface normals were then sent to the RoboDK with TCP/IP connection where the measured normals were paired with the reference normals. In RoboDK, the program calculated the location and orientation of the actual object and gave the position of the 3D-model relative to the robot. This updated position of the object allowed the scanning movements to be performed. The accuracy of the positioning was not examined thoroughly, only found to be sufficient enough.

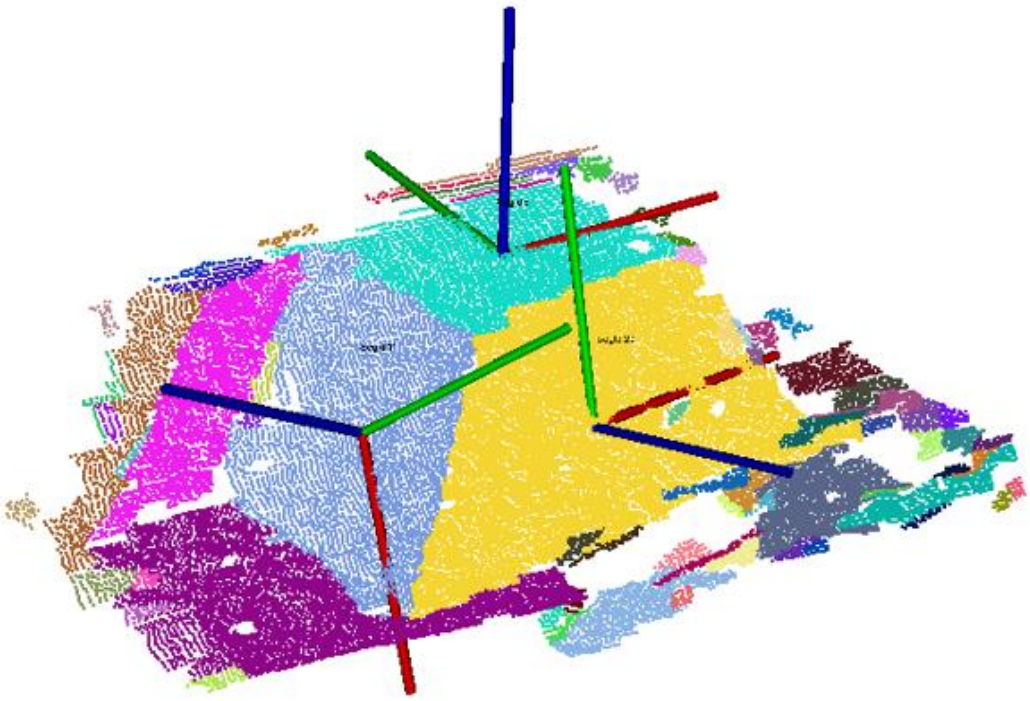

Figure 62. Three segments selected from the filtered and segmented point cloud.

## 7.2 Scanning movements

After the location was updated, scanning movements were programmed in RoboDK and program uploaded into the robot. Both µEpsilon scanners (introduced in 6.1.3) were tested and used without any noticeable differences. Scanning started at lower-end of the seam and moved along the seam to top surface as seen in Figure 63. Different scanning speeds were tested, ranging from 1 cm/s to 10 cm/s but influence of the speed relative to the accuracy was not experimented more thoroughly.

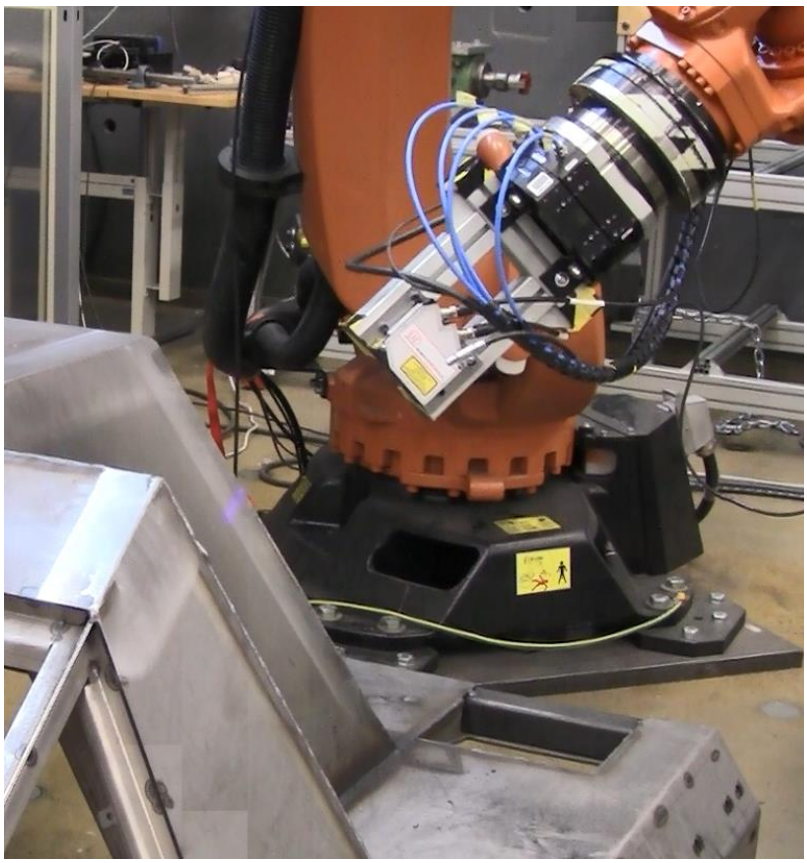

Figure 63. Image from the scanning operation done with 2900-100/BL.

## 7.3 Precise localization

After scanning movements had been performed, the point cloud was segmented with algorithms provided by PCL. The reference normals were provided by RoboDK to the pose estimator, which used the segmented point cloud to localize the object. After the pose had been estimated, the pose was sent back to RoboDK to update the object location in simulation. The segmented point cloud is shown in Figure 64 on the left and on the right is the point cloud fitted into 3D-model by the pose estimator.

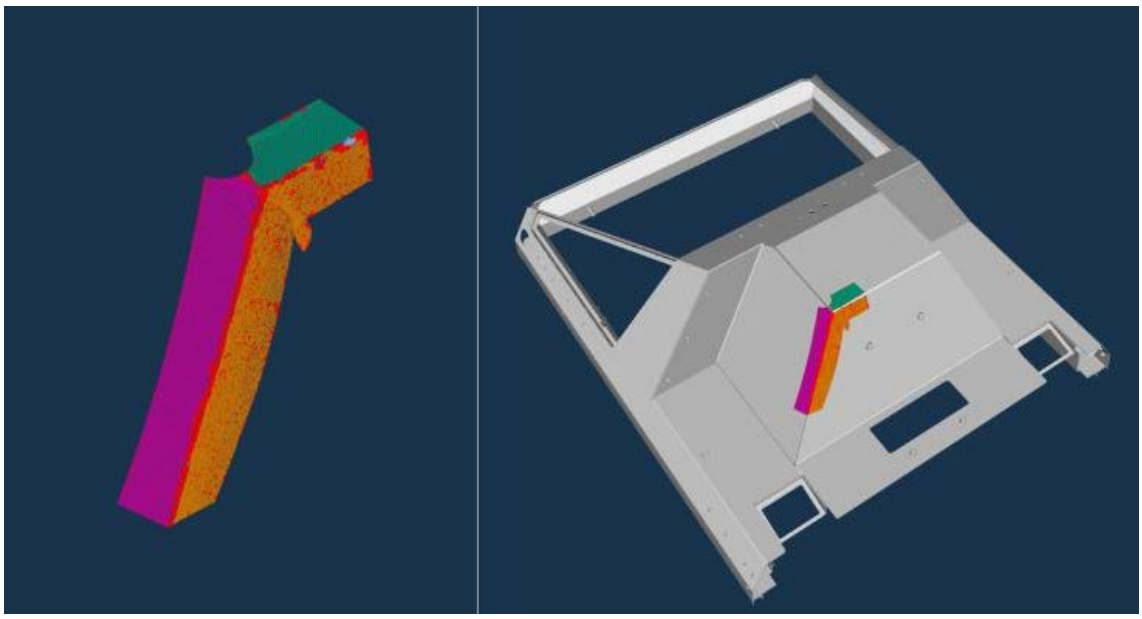

Figure 64. Segmented point cloud fitted into 3D-model.

## 7.4 Seam grinding

For seam grinding, the best grinding sequence was found after multiple experiments. The main challenge for the seam grinding was caused by the irregularities in the weld that led to uneven surfaces even after grinding. After some experimenting, the best structure for the grinding was found to be in two or three phases depending on the desired outcome. For all the situations, sides of the seam were first grinded twice with the angle grinder to remove all the excessive material from the weld. Outcome of this grinding was a sharp corner between the surfaces. After the first phase, the structure depended on the desired roundness of the corner. If the roundness was desired to be greater, some mid-angle vertical grinds with the angle grinder were required to create a bigger radius for the rounding. In the last phase, rounding movements were performed with the sander to smoothen to seam. If the desired outcome was just remove the sharpness of the seam, mid-angle grindings were skipped and the sander alone was enough to round the seam. In Figure 65, on the left is the seam without grinding, on the middle the seam after side grindings and on the right rounding with some mid-angle grinds for greater radius of the corner.

Figure 65. Seam in different phases from left to right: without grinding, after vertical grinding and after rounding movements.

## 7.5 Discussion

For the global localization, main challenge was on finding the best parameters for the segmentation. There are simply no parameter setup, which will work for every situation. Setting the parameters for a certain setup requires an expertise and experience with the parameter setup. After segmentation performs as desired, the localization itself is quite effortless due to the robustness of PCA.

Scanning with 2D-profile scanners proved to be quite straightforward process. The two scanners that were used both had enough long measurement ranges so that data from the both surfaces were obtained by scanning along the seam. The same challenge of setting up parameters for the segmentation exists with the precise localization. However, for our kind of work object, which is composed of similar kind of geometric features, same parameters worked in regardless of the seam that was scanned. Therefore, the objects that consist of similar surface types could theoretically use only one parameter setup.

Force-controlled motions proved to work with the grinding quite well. By analyzing the sensor data, it was proved that the tool remained in contact with the object reliably. The contact force depended a lot from the used speed as well as the regularity of the weld. In the mid-angle grindings where the seam was already grinded, the nominal force deviated

between 15-25 N with set value of 20 N. Accuracy of the force-control could be increased with more accurate calibration and slower grinding speed.

Whereas the speed of vertical grindings was limited only by the force-sensors control system, the experiments showed that rounding speed had a limit that could not be exceeded. The limit was traced to originate from the very characteristics of an industrial robot. It came from the fact that the targets in the rounding are physically so close to each other that the robot has no time to accelerate to target speed. One way to boost the speed was found and it focused on the use of robot's own path planner. The path planner will modify the path by adding slides and approximations so that the robot can run the path in target speed. This way the rounding speed could be increased. However, with our grinding tools, grinding speed could not be increased by much since it had very direct influence on the quality of the grind.

# 8 CONCLUSIONS

The approach of the robot skill programming has been represented as the solution for making the robots easier to program and implement into production. Ideally, the operators in the production line could reconfigure the program for different situations without the need of deeper experience in robot programming. The aim of this thesis was to design and implement skill-based programs, which can be used to localize and grind the work object. These skills would rely heavily into off-line programming methods, because they offer a more efficient programming method and as the 3D-models can be seen as standard in any modern mechanical design process. The thesis started with introducing the essential subjects such as: some basic knowledge of robotics, the principle of offline-programming and theory behind the 3D machine vision methods used in the experiments of this thesis. This was followed by the introduction of skill-based programming concept. In the experimental part of the thesis, structures of the following skills were represented: global localization, scanning with robot, object precise localization, level surface grinding and straight seam grinding. Each skill was designed so, that they are fully independent from the other skills and can be applied easily to other processes.

3D-depth sensors that are mainly aimed to the consumer market can offer surprisingly valid candidates even for the industrial purposes. Limiting factors depend a lot on the size of the area in question, the geometrical properties of the object as well as the accuracy needed. The size of normal 6-DOF industrial robot workspace can be quite easily monitored through a couple of consumer-targeted 3D-depth sensors. Big objects with large plane-like surfaces are the easiest type of objects to get reliable results without the need of fine-tuning the parameters. In this thesis, they were generally used to provide an estimate of the position, for which the required accuracy was not very high.

For the precise localization, 2D-profile scanner attached to the robot's flange was used to localize the object more accurately. The obtained point cloud was then fitted into the 3D-model of the object to provide the location of the object relative to the robot. Accuracy of this kind of localization proved to be more than enough for the grinding purposes, but it required that the scanned area was close to the grinding area. Scanning over one corner will not localize the whole object accurately. If the other seam is located in the opposite side of the object, it will require a new scanning routine closer to the grinding area. The main drawback of the scanning procedure is the time-consuming nature of scanning

movements. It is why scanning the whole object at once is not practical; it is more tempting to localize the areas where the actions will be performed. The localization errors will increase as the distance from the scanned area increases. Same limitations for the objects are present as with the 3D-depth sensors, more complex geometric features are harder to define mathematically and segmentation algorithms to segment reliable repeatedly.

After the object had been localized and the position updated to the robot, different grinding programs were implemented. Even though the object's location was fairly accurately known, force-controlled motions were used in the grindings. The force-control was able to maintain the contact between the tool and the object successfully. Advantages of the force-control were that it helped to encounter localization errors, offered more even grinding for irregular welding seams and enhanced the grinding effect by applying nominal force.

Two grinding skills were developed: level surface grinding and straight seam grinding. Grinding skills were programmed completely in RoboDK, which allowed simulating the grinding path right after the localization had been completed. The simulation notified if the actions were not performable due to issues with the reachability or if the robot's joints were crossing singularities. Both skills required only four input targets, which defined all the movements in the program. The input targets were easily programmed in RoboDK, which made reconfiguring the program fast and effortless. Quality of the grind was altered by the parameters such as grinding speed and nominal force. Grindings were limited to level surfaces and straight seams, mainly because our work object did not have any other geometrical features. The experiments provided an understanding of how the different parameters influence on the grinding quality as well as the limits for the grinding with robot.

As an outcome of this thesis, all the skills were designed and implemented successfully. The objects with plane-like surfaces and available 3D-models can be localized and machined with satisfactory results. The future development should focus on expanding the skills to localize and operate the objects with other geometric properties than just plane-like features. This development requires defining different geometrical shapes mathematically and recognizing the differences between the different shapes. In addition to extending localization and grinding skills to more complex objects, an alternative

method for the objects that have no 3D-models available should be developed. Since grinding skills work with only four given input targets, the targets could be taught the robot with other methods than the traditional online programming. This could mean pointing the targets with a special tool or some other method. These taught targets could then be combined with the localization got from 3D machine vision, so that the robot system will automatically recognize the object and place the targets based on the measurements.

# 9 REFERENCES

Adept Technology, 2007. Robot Configuration Singularities. [online document] http://www1.adept.com/main/KE/DATA/Procedures/Singularity/Singularity.pdf [referred 4.1.2019]. 12 p.

Ahola J., Heikkilä T., Koskinen J., Järviluoma M. & Seppälä T, 2014. Calibration procedures for object locating sensors in flexible robotized machining. IEEE/MESA2014 Conference. DOI: 10.1109/MESA.2014.6935567

ATI Industrial automation, 2010. Six-Axis Force/Torque Sensor System: Compilation of Manuals. 224p.

Basu M., 2002. Gaussian-based edge-detection methods - a survey. IEEE Transactions on systems, man, and cybernetics, Part C: Application and reviews, Vol 32, No 3. p. 9.

Beyerer J., Leon P. & Frese C., 2016. Machine Vision, Automated Visual Inspection: Theory, Practice and Applications. Springer. 798 p. ISBN 978-3-662-47793

Bøgh S., Nielsen O., Pedersen M., Krüger V. & Madsen O., 2012. Does your robot have skills? The 43rd International Symposium on Robotics (ISR2012), Taipei Taiwan. p. 6.

Corke P., 2017. Robotics, Vision and Control. Second edition. Springer. 693p. ISBN 978-3-319-54413-7

Elatta A., Gen L., Zhi F., Daoyuan Y. & Fei L, 2004. An Overview of Robot Calibration. Information Technology Journal 3 (1), p. 74-78. ISSN 1682-6027.

Everett L., Driels M., Mooring B., 1987. Kinematic modelling for robot calibration. IEEE International Conference on Robotics and Automation. DOI: 10.1109/ROBOT.1987.1087818

Felzenswalb P. & Huttenlocher D., 2004. Efficient Graph-Based Image Segmentation. International Journal of Computer Vision 59(2), p. 167-181.

Geng J, 2011. Structured-light 3D surface imaging: a tutorial. Advanced in Optics and Photonics 3, p. 128-160. OCIS Code 150.6910, 110.6880

Gruver W., Soroka B., Craig J. & Turner T., 1984. Industrial robot programming languages: A Comparative evaluation. IEEE Transactions on systems, man and cybernetics vol smc-14, no. 4. p. 565-570.

Heikkilä T. & Ahola J., 2018. Robot skills - modeling and control aspects. IEEE/MESA2018 Conference, Oulu. 6p.

Intel 2019. RealSense Depth Camera D415 technical data. https://click.intel.com/intelr-realsensetm-depth-camera-d415.html [referred 4.1.2019]

ISO 8373, 2012. Robotis and robotic devices - Vocabulary. 2nd edition.

KUKA Roboter Gmbh, 2015. KUKA System software 8.3, Operating and Programming Instructions for System Integrators. 4th edition. 491 p.

KUKA Roboter Gmbh, 2018a. KR 120 R2500 PRO Technical data.

KUKA Roboter Gmbh, 2018b. KUKA.SystemSoftware. https://www.kuka.com/en-de/products/robot-systems/software/system-software/kuka_systemsoftware [referred 4.1.2019]

KUKA Roboter Gmbh, 2018c. KUKA KR C4 [website]. https://www.kuka.com/fi-fi/products/robotics-systems/robot-controllers/kr-c4 [referred 4.1.2019]

KUKA Roboter Gmbh, 2018d. KUKA smartPAD. https://www.kuka.com/en-au/products/robotics-systems/robot-controllers/smartpad [referred 4.1.2019]

Makino H., 1982. Assembly robot. US patent number 4,341,502. 27 July 1982.

Micro-Epsilon, 2019. Laser scanners for 2D/3D profile measurements. https://www.micro-epsilon.com/2D_3D/laser-scanner/ [referred 4.1.2019]

Nguyen A. & Le B., 2013. 3D Point Cloud segmentation: A survey. Conference paper. DOI: 11.1109/RAM.2013.6758588

Omron, 2016. SCARA Robots: Cobra 350.
http://www.omron.com.au/products/family/3515/dimension.html [referred 4.1.2019]

Pan Z., Polden J., Larkin N., Van Duin S. & Norrish J., 2011. Recent progress on programming methods for industrial robots. Robotics and Computer-Integrated Manufacturing 28, p. 87-94.

Pedersen M., Nalpantidis L., Andersen R., Schou C., Bøgh S., Krüger V., Madsen O., 2016. Robot skills for manufacturing: From concept to industrial deployment. 2016. Robotics and Computer-Integrated Manufacturing 37 2016, p. 282-291.

Perez L., Rodriguez I., Rodriguez N., Usamentiaga R., Garcia D. Robot guidance using machine vision techniques in industrial environments: a comparative review. Sensors 2016, 16, 335. p. 26.

RoboDK, 2018. Simulation and OLP for Robots [website]. www.robodk.com [referred 4.1.2019]

Salvi J., Pages J. & Batlle J., 2004. Pattern codification strategies in structured light systems. Pattern Recognition 37, p. 827-849.

Schnabel R., Wahl R. & Klein. R., 2007. Efficient RANSAC for Point-cloud shape detection. Computer graphics forum, Volume 26, 2007, Number 2, p. 214-226.

Shiakolas P., Conrad K. & Yih T, 2015. On the Accuracy, Repeatability and Degree of Influence of Kinematics Parameters for Industrial Robots. International Journal of Modelling and Simulation, 22:4, p. 245-254. DOI: 10.1080/02286203.2002.11442246

Shlens J., 2014. A Tutorial on Principal Component Analysis. Google Research. arXiv:1404.1100v1 [cs.LG]. p. 12.

Siciliano B., Khatib O., 2016. Handbook of Robotics. 2nd edition. Springer, 2227 p. ISBN 978-3-319-32550-7

Tsai R., Lenz R., 1989. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. IEEE Transactions on robotics and automation. Vol. 5, No. 3.

Yamaha Motor Co. 2018, Cartesian robots (Application examples). https://global.yamaha-motor.com/business/robot/lineup/application/xyx/index.html [referred 4.1.2019]

Wallen J., 2008. The history of the industrial robot. Technical report from Automatic Control at Linköpings Universitet. LiTH-ISY-R-2853. 16 p. ISSN: 1400-3902.

# APPENDIX

Appendix 1. Flowchart of the grinding with localization -task.

Appendix 1. Flowchart of the grinding with localization -task.