

Samu Rautiainen

**DESIGN AND IMPLEMENTATION OF A  
MULTIMODAL SYSTEM FOR HUMAN-  
ROBOT INTERACTIONS IN BIN-PICKING  
OPERATIONS**

Master of Science Thesis  
Faculty of Engineering and  
Natural Sciences  
Jose Martinez Lastra  
Luis Gonzalez Moctezuma  
August 2022

# ABSTRACT

Samu Rautiainen: Design and Implementation of a Multimodal System for Human-Robot Interactions in Bin-Picking Operations  
Master of Science Thesis  
Tampere University  
Master's Degree Programme in Automation Technology  
August 2022

---

Manufacturing domains are evolving into *Industry 4.0*, which requires design choices that leads to flexible, modular, and human-centred solutions. In robotics this phenomenon is seen as utilizing the human flexibility in solutions, in other words solving robotic problems by using the Human-Robot Collaboration. Successful collaboration is achieved with intuitive Human-Robot Interaction methods.

The most efficient way to communicate with a robot is to use same communication methods that is seen in human-human communication, such as gaze, speech, touch, and body gestures. Information exchange of two humans communicating consists of mostly body language and most of the activities that the human does in their life are done by using their hands. Therefore, hands present as an excellent source for the input when interacting with a robot.

Bin-Picking is seen everywhere, from warehouses to manufacturing processes, and is mostly done by human. Automating such process with robot has endless applications and would lead in higher productivity. Bin-Picking as a robotic problem does not have universal solution, and most of the solutions relies on the fact that a 3D model of the parts that needs to be picked is known. With unknown parts, the systems usually set limitations to the shape of the objects.

The main goal of this Thesis is to design and implement a system and a process, that creates a new use for collaborative robot cell, and teaches the system to be able to do Bin-Picking operations automatically with unknown parts. Furthermore, the proposed solution will fit to the so called "Smart Factory", which is considered as fundamental concept of *Industry 4.0*.

The solution consists of four major components, Orchestrator Application, Robot Controlling Application, 3D-module that finds grasp poses for unknown parts and M2O2P that reads hand gestures from smart glove sensor input. The solution is evaluated in component basis, and in a process working together. Furthermore, the taught system is tested in process created for the evaluation, where the robot picks parts from a bin and hands them to the human operator.

Keywords: Human-Robot Interaction, Human-Robot Collaboration, Collaborative Robot, Bin-Picking, Smart Factory, Industry 4.0

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

# TIIVISTELMÄ

Samu Rautiainen: Multimodaalisen järjestelmän suunnittelu ja toteutus ihmisen ja robotin väliseen kanssakäymiseen automatisoidun poiminnan operaatioissa

Diplomityö

Tampereen yliopisto

Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma

Elokuu 2022

---

Teollisuus on kehittymässä kohti *Industry 4.0* -konseptia, ja täten vaatii suunnitteluvalintoja, joiden ansiosta voidaan tuottaa joustavia, modulaarisia ja ihmiskeksisiä ratkaisuja. Robotiikassa tämä muutos nähdään ihmisen joustavuuden hyödyntämisenä toteutuksissa, eli robotiikan ongelmien ratkaisemista hyödyntäen ihmisten ja robottien välistä yhteistyötä. Onnistunut yhteistyö voidaan saavuttaa käyttämällä intuitiivisia ihmisten ja robottien väliseen kanssakäymiseen käytettyjä menetelmiä.

Tehokkain tapa kommunikoida robotin kanssa on käyttää samoja kommunikaatiomenetelmiä, mitä käytetään ihmisten välisessä kanssakäymisessä, kuten katsetta, puhetta, kosketusta tai kehon elekieltä. Ihmisten välisessä kommunikaatiossa tiedonvaihto tapahtuu pääosin kehon elekielillä, ja ihmiset käyttävät käsiään suurimpaan osaan aktiviteeteista heidän elämässään. Täten kädet toimivat loistavasti lähteenä robotin kanssa kommunikoimiselle.

Laatikoista poimimista nähdään kaikkialla, aina varastoista teollisuusprosesseihin, ja se tehdään pääosin ihmisten toimesta. Tällaisten prosessien automatisointiin on olemassa loputtomasti käyttökohteita, ja automatisointi johtaa korkeampaan tuottavuuteen. Automaattiseen poimintaan robottien avulla ei ole olemassa universaalia ratkaisua, ja monet ratkaisut pohjautuvat ennakkotietoon poimittavien osien 3D malleista. Tuntemattomilla osilla automaattisen poiminnan järjestelmät yleensä asettavat rajoitteita poimittavien osien muodolle.

Tämän diplomityön tavoitteena on suunnitella ja toteuttaa järjestelmä ja prosessi, joka tuottaa uuden käyttötarkoituksen yhteistyörobottisolulle, ja opettaa järjestelmälle, kuinka tehdä automaattisen poiminnan operaatioita tuntemattomilla osilla. Lisäksi ehdotettu toteutus sopii älykkääseen tehtaaseen, joka on yksi olennaisista *Industry 4.0* -konsepteista.

Ratkaisu koostuu neljästä suuremmasta komponentista, orkestrointi sovelluksesta, robotin ohjaussovelluksesta, 3D-moduulista, joka löytää poiminta-asennon tuntemattomille osille, sekä M2O2P komponentista, jolla luetaan älykkäällä hanskalla tehtäviä käsimerkkejä. Ratkaisu arvioidaan komponentti kerrallaan, ja prosessissa, jossa komponentit toimivat yhdessä. Lisäksi opettettu järjestelmä testataan arviointia varten tehdyllä prosessilla, jossa robotti nostaa osia laatikosta ja ojentaa ne ihmisoperaattorille.

Avainsanat: Ihmisen ja robotin välinen kanssakäyminen, ihmisen ja robotin yhteistyö, yhteistyörobotti, automaattinen poiminta, älykäs tehdas, *Industry 4.0*

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

## PREFACE

Firstly, I want to thank for the help of Professor Jose Martinez Lastra and Luis Moctezuma Gonzalez of supporting my journey through the Thesis process. Additionally, I wouldn't have succeeded in my Thesis in the same extend that I did without the help of my colleague, Seyedamir Ahmadi.

Secondly, I would like to thank my beloved life partner Jenni, who helped me to push my best in the Thesis process. She was the closest supportive person I needed to have to make it through the Master of Science degree. Without her to listen to me describing my problems in my studies, which she didn't even understand, would have prevented me to get so many revelations.

Lastly, I want to thank all my friends I have found as result of my university studies. These people have helped me to push through the master's degree and provided me peer support when it was needed.

Tampere, the 14<sup>th</sup> of August 2022

Samu Rautiainen

# CONTENTS

1.INTRODUCTION.....	1
1.1 Motivation .....	1
1.2 Problem Statement and Research Questions.....	2
1.3 Objectives .....	2
1.4 Limitations.....	3
1.5 Background.....	3
2.LITERATURE REVIEW.....	5
2.1 Smart Factory .....	5
2.2 Human-Robot Interactions and Collaboration.....	6
2.2.1 Safety .....	9
2.2.2 Control and Robot Programming.....	11
2.3 Natural human interaction methods.....	12
2.4 Bin-Picking.....	14
2.4.1 Point Cloud processing .....	16
3.DESIGN.....	17
3.1 Components.....	17
3.2 Solution Design.....	20
3.3 Architecture and Communication .....	23
4.PROPOSAL .....	27
4.1 Architecture and Communication .....	27
4.2 M2O2P.....	29
4.3 Orchestrator Application.....	33
4.4 Robot Control.....	33
4.5 3D-Module .....	34
5.IMPLEMENTATION AND TESTING.....	35
5.1 M2O2P.....	35
5.1.1 CaptoGlove SDK.....	35
5.1.2 Application Controller .....	36
5.1.3 Web User Interface .....	40
5.1.4 ROS2-FIWARE bridge .....	42
5.1.5 Integration Service .....	43
5.2 Orchestrator Application.....	43
5.3 Robot Control.....	45
5.3.1 ABB Robot Controller.....	45
5.3.2 Robot Controlling Application .....	46
5.4 3D-module .....	47
5.4.1 Acquire PCD .....	47
5.4.2 Calibration.....	48
5.4.3 Find grasp pose .....	51

5.5	Testing Procedures for the System .....	57
5.5.1	Isolated tests for M2O2P.....	58
5.5.2	Isolated tests for 3D-module .....	59
5.5.3	Tests done for complete Use-Case .....	60
6.	RESULTS .....	63
6.1	Isolated tests for M2O2P .....	63
6.2	Isolated tests for 3D-module .....	65
6.3	Tests done for complete Use-Case .....	68
6.3.1	Tests done in the teaching process.....	68
6.3.2	Tests done in the evaluation process.....	72
7.	CONCLUSIONS.....	76
7.1	Future work.....	77
8.	REFERENCES .....	79

## LIST OF FIGURES

<b>Figure 1.</b>	<i>Number of research papers covering HRI and HRC.....</i>	<i>6</i>
<b>Figure 2.</b>	<i>Collaboration levels and how the environment is shared [26].....</i>	<i>8</i>
<b>Figure 3.</b>	<i>Levels of Human-Robot Collaboration [26].....</i>	<i>8</i>
<b>Figure 4.</b>	<i>Elements to ensure safe HRI [34].....</i>	<i>10</i>
<b>Figure 5.</b>	<i>Cross section of finding grasp position without colliding to other parts.....</i>	<i>15</i>
<b>Figure 6.</b>	<i>On robot two finger gripper [76], Robotiq vacuum gripper [77] and ABB smart gripper [78] with vacuum and two fingers combined .....</i>	<i>15</i>
<b>Figure 7.</b>	<i>Locations of the sensors, on left palm side down picture the bending sensors and pressure sensors in right palm side up picture are presented in blue color and the transmitter is presented in orange color .....</i>	<i>18</i>
<b>Figure 8.</b>	<i>ABB Yumi Robot [91] .....</i>	<i>18</i>
<b>Figure 9.</b>	<i>Zivid One+ 3D camera [94] .....</i>	<i>19</i>
<b>Figure 10.</b>	<i>The cell collaborative robotic cell layout .....</i>	<i>19</i>
<b>Figure 11.</b>	<i>Process flow of the teaching process .....</i>	<i>22</i>
<b>Figure 12.</b>	<i>General architecture of the system and components within it, and their connections to FIWARE. ....</i>	<i>23</i>
<b>Figure 13.</b>	<i>Example of the Task entity .....</i>	<i>24</i>
<b>Figure 14.</b>	<i>Example of the Device entity.....</i>	<i>25</i>
<b>Figure 15.</b>	<i>Example of the Subscription entity .....</i>	<i>26</i>
<b>Figure 16.</b>	<i>Architecture diagram of the solution .....</i>	<i>27</i>
<b>Figure 17.</b>	<i>Communication between the components.....</i>	<i>28</i>
<b>Figure 18.</b>	<i>Use-Case diagram of human operator interacting with M2O2P component.....</i>	<i>29</i>
<b>Figure 19.</b>	<i>Software Architecture of M2O2P.....</i>	<i>31</i>
<b>Figure 20.</b>	<i>Sequence diagram of M2O2P receiving and completing a task.....</i>	<i>32</i>
<b>Figure 21.</b>	<i>The format of the message sent from SDK to AC.....</i>	<i>36</i>
<b>Figure 22.</b>	<i>Sensor readings of each finger and two gestures done within 20 seconds time window.....</i>	<i>37</i>
<b>Figure 23.</b>	<i>Flowchart of AC sensor data processing .....</i>	<i>38</i>
<b>Figure 24.</b>	<i>ROS2 nodes and topics communicating in M2O2P .....</i>	<i>39</i>
<b>Figure 25.</b>	<i>Monitoring section of the GUI.....</i>	<i>40</i>
<b>Figure 26.</b>	<i>Calibration section of the UI .....</i>	<i>41</i>
<b>Figure 27.</b>	<i>Testing and additional options section.....</i>	<i>42</i>
<b>Figure 28.</b>	<i>Integration Service architecture [110].....</i>	<i>43</i>
<b>Figure 29.</b>	<i>Process in JSON format given to Orchestrator Application in teaching process.....</i>	<i>44</i>
<b>Figure 30.</b>	<i>Process in JSON format given to the taught system to validate 3D-module.....</i>	<i>45</i>
<b>Figure 31.</b>	<i>Example of 2D image captured by the Zivid 3D camera.....</i>	<i>48</i>
<b>Figure 32.</b>	<i>Workspace from top view with chessboards in the corners .....</i>	<i>48</i>
<b>Figure 33.</b>	<i>Calculating the index of the corner point in point cloud.....</i>	<i>49</i>
<b>Figure 34.</b>	<i>The middle point (on green color), side points (on red color) and the 3D vectors between the middle point and the side points (in purple color).....</i>	<i>49</i>
<b>Figure 35.</b>	<i>Finding the base tag frame using the camera frame.....</i>	<i>50</i>
<b>Figure 36.</b>	<i>The camera frame, chessboard frame, robot base frame and the point cloud transformed to the robot base frame .....</i>	<i>51</i>
<b>Figure 37.</b>	<i>Downsampled point cloud without the ground plane.....</i>	<i>52</i>
<b>Figure 38.</b>	<i>Found clusters painted with their corresponding colour.....</i>	<i>53</i>
<b>Figure 39.</b>	<i>Line composed of the boundary points for a part.....</i>	<i>54</i>

<b>Figure 40.</b>	<i>The gray points are the boundary points, blue points are non-boundary points, the green dashed line acts as neighbour radius, green point as the calculated mean, green arrow as the vector that drawn from the mean to the boundary point and red arrows illustrates the normal aligned with the vector.....</i>	<i>55</i>
<b>Figure 41.</b>	<i>The gray box is the part, the green point is the boundary point, the purple arrow is the negative of the normal for that point, purple dashed arrow is the same vector but taken next to the yellow candidates for the evaluation of the angle, similarly the yellow dashed lines are taken for the evaluation. The lower candidate would present as aligned pair, but the top candidate doesn't, due the angles that they have .....</i>	<i>56</i>
<b>Figure 42.</b>	<i>Collision avoidance method .....</i>	<i>57</i>
<b>Figure 43.</b>	<i>Example grasp pose found for the robot.....</i>	<i>57</i>
<b>Figure 44.</b>	<i>Parts used in evaluation of the Bin-Picking operations .....</i>	<i>58</i>
<b>Figure 45.</b>	<i>Cutkosky's grasp taxonomy adopted with testing objects.....</i>	<i>59</i>
<b>Figure 46.</b>	<i>Process created for evaluation.....</i>	<i>61</i>
<b>Figure 47.</b>	<i>Results of tests done using the Cutkosky's grasp taxonomy when wearing the smart glove .....</i>	<i>63</i>
<b>Figure 48.</b>	<i>Examples of the manipulation procedures.....</i>	<i>64</i>
<b>Figure 49.</b>	<i>No collision free points found with the example of robot finger colliding on the left side, and found grasp pose on the right side.....</i>	<i>67</i>
<b>Figure 50.</b>	<i>Bin casting a shadow on the chessboard tag .....</i>	<i>67</i>
<b>Figure 51.</b>	<i>Part that caused noisy point cloud due the reflection of metallic side .....</i>	<i>68</i>
<b>Figure 52.</b>	<i>Histogram of interaction times compared to whole human tasks in the process .....</i>	<i>71</i>
<b>Figure 53.</b>	<i>Illustration of teaching procedure using the glove and the mouse and GUI as input method .....</i>	<i>72</i>
<b>Figure 54.</b>	<i>Slope of the part causing motion supervision error from the robot controller.....</i>	<i>74</i>
<b>Figure 55.</b>	<i>Example of the grasp positions with a failed grasping in last image .....</i>	<i>75</i>



## LIST OF SYMBOLS AND ABBREVIATIONS

ABB	Technology company
AC	Application Controller
AI	Artificial Intelligence
API	Application Programming Interface
CPS	Cyber-Physical System
DB	Database
DBSCAN	Density Based Scanning, segmentation algorithm
DoF	Degrees of Freedom
FAST-laboratory	Future Automation Systems and Technologies laboratory
GIF	Graphics Interchange Format
GUI	Graphical User Interface
GUI	Graphical User Interface
HCI	Human Computer Interface
HRC	Human Robot Collaboration
HRI	Human Robot Interaction
HTTP	Hypertext Transfer Protocol
IMU	Inertial Measurement Unit
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
KDtree	K-nearest neighbours tree
M2O2P	Multi-Modal Online and Offline Programming solution
ML	Machine Learning
OA	Orchestrator Application
OCB	Orion Context Broker
PbD	Programming by Demonstration
PC	Personal Computer
PCD	Point cloud
PFL	Power and Force Limiting
PM	Process Management
RANSAC	Random Sample Consensus algorithm
RANSAM	Random Sample Matching algorithm
RCA	Robot Controlling Application
RGB	Red-Green-Blue colours for each pixel
RGB-D	Red-Green-Blue colours and depth value for each pixel
ROS	Robot Operating System
SDK	Software Development Kit
SHOP4CF	Smart Human Oriented Platform for Connected Factories
SRMS	Safety-Rated Monitored Stop
SSM	Speed and Separation Monitoring
TCP/IP	Transmission Control Protocol / Internet Protocol
UI	User Interface
UML	Unified Modeling Language
URN	Uniform Resource Name

# 1. INTRODUCTION

Human-Robot Interactions (HRI) is highly researched field of robotics, and in recent decade there have been growing interest on natural human interaction methods. In a collaborative setup natural input methods makes the communication accessible even for untrained operator. However, for successful collaboration of human and robot, other matters such as the level of the collaboration and safety needs to be considered. Often, the key factor for solving a complex robotic problem is to have human interact with the robot in a way, that the complex tasks of the process are solved by human.

This chapter is divided in five subchapters. The motivation for the research is presented in Sec. 1.1, the main problem that the Thesis tries to solve and research questions in Sec. 1.2, objectives of the Thesis in Sec. 1.3, limitations for the implementation in Sec. 1.4 and background of the Thesis in Sec. 1.5.

## 1.1 Motivation

The demand for mass customization of products have created requirements for manufacturing to evolve from automated and mass production manufacturing lines to modular and flexible solutions, to *Industry 4.0* [1]. In a robotic scheme, this evolvment means the necessity of exploiting collaborative robots instead of traditional industrial robots [2]. The International Federation of Robotics reported that the amount of installed collaborative robots annually is clearly increasing; approximately 11 000 collaborative robots were installed in year 2017, whereas in 2020 the number was 22 000 [3].

Successful collaboration requires communication between the actors, and from human perspective it is beneficial to be able to communicate with the robot with natural input methods. When the communication is established in similar manner as human-human communication, it will be intuitive and approachable for the human. To develop human-robot interface for natural human inputs is not an easy task and has many open questions about the reliability and overall feasibility of such system [4].

Bin-Picking operations are seen extensively in industrial domains, and are mostly done by humans [5]. Universal Robots reported that in the United States, 38% of workers in manufacturing moves parts from bins to machines [6]. By automating such actions with the help of robots, the systems will provide higher efficiency and saves the human worker

from laborious tasks. The Bin-Picking robotic problem is well researched, yet no universal solution exists for the problem [7]. Many of the solutions assumes that the 3D model of the parts that are picked from the bin is known (e.g., [8], [9]). When the parts are unknown, additional data processing methods are required to be developed for finding the essential information for successful Bin-Picking.

## 1.2 Problem Statement and Research Questions

Robot cells should be able to be reused modularly and flexibly to answer the needs of mass customization and the *Industry 4.0*. This thesis tries to find solution to such problem through the design and utilization of Human-Robot Collaboration (HRC) enabled with HRI.

The questions about how the system can be created and how it performs shape the research questions as follows:

- How to implement gestural HRC based system for industrial applications?
- How to create a component that processes 3D data for Bin-Picking operations?
- How gestural HRI method compares to more traditional method using GUI?

The first question includes multiple aspects and will be covered through the whole Thesis. To answer to the second question, the required information about the Bin-Picking and 3D data processing are presented in Literature Review, and the Proposal and Implementation and Testing -chapters provides the answer about how such component was created. The last question will be answered through the tests of the Thesis.

## 1.3 Objectives

This thesis aims to finds a solution to the problem of reusability of a collaborative robot cell for Bin-Picking operations and simultaneously answer to the research questions. To create such solution, required design aspects are researched and presented through literature, and considered when designing and implementing the solution. In this context, reusability is approached by teaching new information for the system with the help of human, which can then be invoked later in the same or another process. In this thesis the state of the system after the teaching is referred to as “taught system”. Implementation-wise the objectives can be listed as follows:

- Design a system architecture that complements the concept of Smart Factory

- Design a process that can be used to solve the reusing of the collaborative robot cell
- Implement an orchestrator that can assign the tasks to other components in the solution
- Design and implement a component for HRI
- Implement a component that handles communication with a collaborative robot
- Implement modular procedures for the robot controller
- Implement a component that can communicate with 3D camera and process the 3D data

When the implementation is ready, the last objective is to evaluate the solution and conclude the Thesis.

## 1.4 Limitations

Limitations for this Thesis were identified partly at the start of the Thesis implementation and partly during it. Limitations are as follows:

- Implementation tries to find a solution to Bin-Picking with the help of the human operator, which is possible only if the used robot supports collaborative tasks
- Smart glove reading application works only with CaptoGlove and changes are needed if there is a need to use another input device
- The used collaborative robot has restrictions on carry weight, which limits the parts that the solution can effectively pick up
- 3D-module will handle only objects that are formed mainly from 90-degree angles, that are picked always from top and are semi-structured, which limits the possible parts and their poses that the system can handle

## 1.5 Background

The Multimodal Offline and Online Programming (M2O2P) solution -component, which main function is to read and process the smart glove data, was implemented in the Smart Human Oriented Platform for Connected Factories (SHOP4CF) EU-project. SHOP4CF is a Horizon 2020 project, that aims for human-centric manufacturing and to create Smart Factory components that operate in highly connected factor model [10].

The solution created as outcome of this Thesis utilizes architecture and data models of the SHOP4CF. The solution is then compatible with other SHOP4CF components by design and utilizes pre-existing interfaces what was created for M2O2P component. Furthermore, the design and implementation of M2O2P is presented in this Thesis.

## 2. LITERATURE REVIEW

In this chapter the major topics of the Thesis will be presented through literature. Firstly, the review will focus on concept of Smart Factory in Sec. 2.1 The Sec. 2.2 will focus on HRI and HRC in general, safety aspects (Sec. 2.2.1) of them and the robot control and programming paradigms that are important in HRI/HRC. The Sec. 2.3 will present natural human input methods, with the focus on the hand gestures. Lastly the Bin-Picking operations are presented in Sec. 2.4, where the review goes through the problem and possible methods for solving it.

### 2.1 Smart Factory

One of the fundamental concepts brought up by *Industry 4.0* are so called “Smart Factories” [1]. Smart Factory connects the entities, such as sensors, actuators, and computers in the factory to same network and with data computing tools achieves a status of smart environment [11]. The definition describes modular and connected shopfloor, that solves a dynamically changing conditions of production by being adaptive and flexible [12]. Smart Factory utilizes the human flexibility and is referred as human-centred [13].

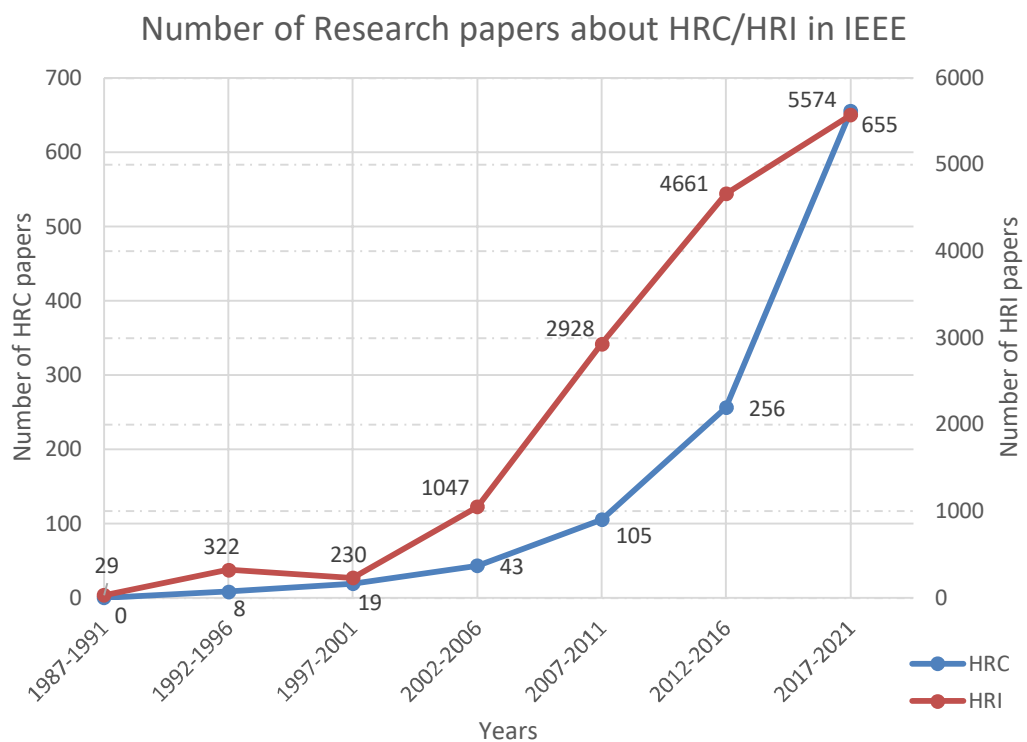
From technical standpoint, Smart Factories are connected factories, that consist of context-aware systems and entities and aim to assist the task execution of humans and machines within the factory [14]. Context information of Smart Factory component represents the physical object or application information such as condition or position, that are gathered with sensors or provided by the component itself. When applications and systems can update and consume context information, they are described as context-aware [14]. By having an architecture consisting of context-aware components, all the applications will have common medium and can modularly connect to each other over the limits of traditional automation hierarchy [15]. The systems that unite the physical real-life actors and communication and computing of digital world are called Cyber-Physical System (CPS) [14]. CPS enabled with Internet of Things (IoT) provides interface for the context information to the computing applications by connecting the entities with internet [16].

To answer to the requirements of Smart Factory, robotic applications demand the human to provide the flexibility for the solution. The need for HRI and HRC is therefore apparent.

## 2.2 Human-Robot Interactions and Collaboration

HRI as a field of research covers the study of interactions and communication between one or more humans and one or more robots. These interactions are present in all of robotics since in all robotic applications robots are working for and used by humans. HRI can be described as interdisciplinary, and can be seen as combination of natural sciences, engineering, cognitive sciences and psychology, and human factors [17].

HRI is required for having a functional Human-Robot Collaboration (HRC) system. The manufacturing is heading in direction of collaborative robotics due the flexibility and performance increment such systems offer [18]. HRI and more recently the HRC have growing interest as a field of studies. One well speaking metric of this is the research articles written about them. Using IEEE Explorer with “Human-Robot Interaction” and “Human-Robot Collaboration” as keywords, following graph was created (Figure 1). The graph has two y-axes to combine the graphs.



**Figure 1.** Number of research papers covering HRI and HRC

Robots are frequently thought as passive machines, that does repetitive work with high precision, yet are limited by the programming of the robot controller and will not succeed if new unknown task is presented [19]. If there is an ability for human to give guidance for the robot, the behaviour of the robot becomes more flexible. Human advantages,

such as flexibility and ability to handle complex tasks, coupled with robot's reliability and robustness, presents an efficient team [20].

Depending on location of human and robot, the interaction can be remote or proximate [17]. Remote interactions can happen when human and the robot are separated by location, for example when robot is located in hazardous environment where human cannot enter (e.g. [21]) or additionally separated with time (e.g. [22]). Proximate interaction describes a system where the human and the robot are in same space, for example in the case of service robots (e.g. [23]) or collaborative robots (e.g. [24]).

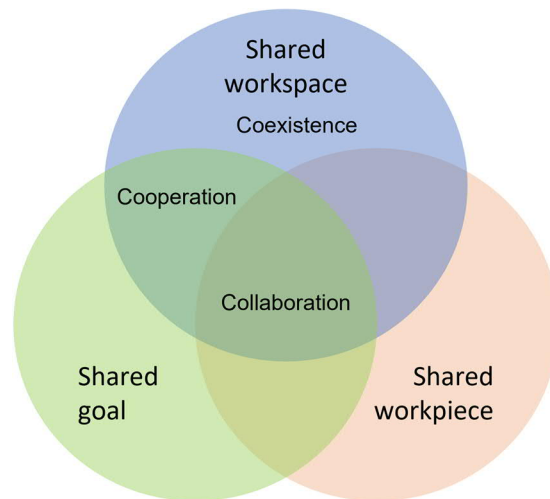
When designing an HRI system, there are five attributes that the designer can affect. These attributes are presented in the Table 1 with explanations.

Table 1. *Attributes that are designer to choose when designing HRI system [17]*

Attribute	Explanation
Level of Autonomy (LOA)	From computer/machine offering no assistance and human doing all the work to computer doing everything and ignoring human [25]
Nature of information exchange	The communication format and medium between the human and robot, such as visual (e.g., displays), body gestures, speech, alert sounds and physical interaction
Structure of the team	The number of humans and robots teamed
Adaptation, learning and training	How to train the human for HRI, how to adapt to new environment and how the people, and robot are learning from the interactions
Shape of the task	How the task will and should be done when new technology is introduced

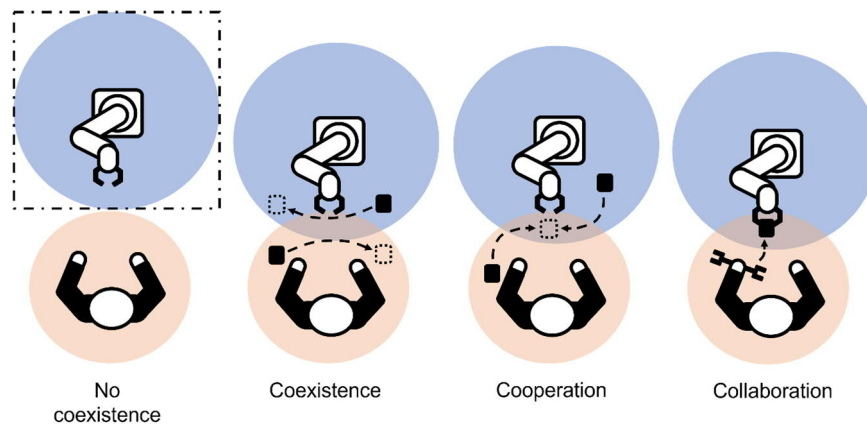
These attributes cover only the field of HRI. When the robot and human are collaborating, there are further design metrics available. One of the most important choices for the collaborative work is to determine the level of collaboration. This choice has major influence on the required safety measures and workflow of the system. In this Thesis, the proposed HRC levels by Aaltonen et. al. [26] are adapted. There are four levels, where the first one is no coexistence. This collaboration level refers a system where human, and robot will not share workspace, goal, nor the workpiece. Rest of the collaboration levels, coexistence, cooperation, and collaboration shares their environment. These three levels and how they share the environment are illustrated in Figure 2.





**Figure 2.** Collaboration levels and how the environment is shared [26]

The figure explains how the level of the collaboration increases, due the fact that more paradigms within the collaborative system is shared amongst human and the robot. Figure 3 further explains what these levels mean in practice. The level of collaboration raises from left to right.



**Figure 3.** Levels of Human-Robot Collaboration [26]

The first collaboration level includes the traditional robot work cells. This means that the human cannot enter the workspace. However, the collaborative scenario can be achieved with for example rotary table (low level in [27]), for providing materials to the robot cell.

The coexistence describes a situation, where robot and the human are sharing a workspace but do not have shared goal nor workpiece. This sort of collaboration can be for example human fetching a workpiece from robot work area, or human and robot using same bucket of liquid (medium level in [27]) that is in the shared workspace.

Cooperation between human and the robot explains higher level of collaboration. Example cases could be walk-through of the robot by human hand or with teach pendant, human entering the welding workspace to turn the product and leaving and in the meantime the robot is stopped, or human setting up more parts for the robot to work on [28].

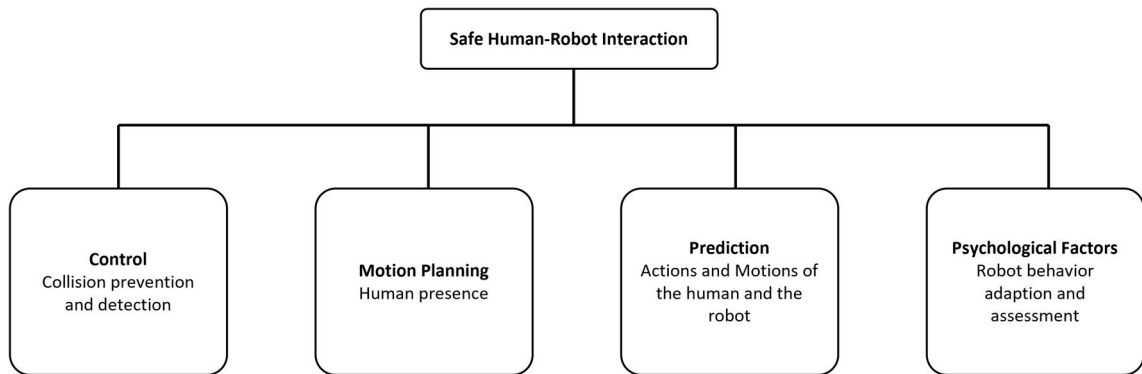
The highest level, collaboration, shares a workpiece and in most advanced settings, works human-like or reads proactively human intentions, such as in [29]. Collaborative setup could be tasks where robot or human holds a part, and the other party member works on the part [30].

### **2.2.1 Safety**

In science fiction novel *Runaround* [31] Isaac Asimov presented three laws of robotics. The first one, “A robot may not injure a human being or, through inaction, allow a human being to come to harm”, is an intuitive and axiomatic rule that can be applied to non-fiction robotic applications.

HRC applications need special acknowledgement on safety measures due the fact that human and robot are sharing workspaces. The need has been taken into account in the form of standards, such as ISO 10218-2 [32], that considers safety requirements for industrial robots, and ISO/TS 15066 [33], that is a technical specification for collaborative robots.

Since the HRC system includes HRI, the safety measures can be explained through the safety in HRI. As was proposed in [34], it can be divided in two categories, physical safety, and psychological safety. Physical safety includes prevention of and reaction to unwanted contacts between human and robot. Psychological safety covers stressors inflicted by the robot with its behaviour to human. The physical safety can be further divided to three categories: control, motion planning, and prediction. Figure 4 presents a tree diagram consisting of these elements, and examples of methods that achieves these set elements.



**Figure 4.** Elements to ensure safe HRI [34]

The safe HRI through control will include collision avoidance and collision detection. Most of the real-time collision avoidance methods are based on Artificial Potential Fields [35]. The algorithm defines attractive force for robot motion, and repulsive forces for objects in the robot's environment. By summing the forces, the algorithm provides dynamical collision avoidance method. The same method has been further elaborated with depth data [36]. In ISO/TS 15066 [33] (technical specification of standard) there are four operational modes shown for HRC. The first one, Safety-Rated Monitored STOP (SRMS), describes a system that stops the robot when human is present in the robot workspace. In addition to this, it is possible to combine the SRMS with Hand Guiding (HG, earlier in the Thesis referenced as walk-through programming), which is triggered if human enters the workspace of the robot and turned off when the human leaves [33]. The system named SafeMove [37] which includes safety zones that can control robot speed, were developed by technology company ABB. The system uses external sensors for tracking the occupancy of human or other objects in the safety zones. In [38] the authors proposed a dynamically changing safety space, that is projected top down to the table. In other words, the safety zone was implemented in x-y-plane.

The collision detection methods aim for detecting the happened collision between the robot and the environment as fast as possible. The collision can be detected by looking for fast transients in robot electrical drives [39], comparing the commanded robot torques to the nominal robot model with disturbance observer [40] or to measured torques if such sensors exist, or using sensitive skins on the robot [41]. When collision is detected, the robot cannot stop and press against the environment. Usually there are post-collision methods that determines the robot actions after the collision, such as backing away from the collision point with same trajectory that the robot reached the collision point in ABB Collision Detection system [42]. The safety of the human operator when collision happens can be achieved with Power and force limiting (PFL) [33], which permits touch

among the human and the robot, but the with the use of force and torque limiting achieves the safe collaboration. However, this is primarily limited to collaborative robots.

The safety in HRI can also be achieved using a motion planning as a premise to make safe environment. Dynamic solution to this approach would be standard operational mode; speed and separation monitoring (SSM, ISO/TS 15066) [33], which will adjust the robot speed regarding the distance between the human and the robot. Another example of such approach would be limiting the jerk and acceleration of the robot when human is present [43].

Collision prediction systems by predicting human activity, such as in [44], can offer a powerful tool to prevent the collisions. Human activity can be tracked through sensors, for example using RGB or RGB-D data. The prediction can be done also predicting human motion, such as in [45] the authors have used time series classification for this purpose.

Different features and actions of robot can cause distress to the human operator. By adapting the robot behaviour for specific user, the system can avoid negative psychological emotions on the human operator. Good example of this is an adaptation of speed of the robot using electroencephalography sensors to read the emotions of the human when collaborating with the robot [46].

## **2.2.2 Control and Robot Programming**

Robot control is a wide and deep topic and therefore the sub-topics of the robot control needs to be narrowed down for this Thesis to purposefully explain the necessary paradigms that can be put under the topics of HRI/HRC. Following paragraphs will explain these topics by providing broad overview on the topic and additionally explains collaborative robot programming paradigms that can be utilized in the implementation.

From design perspective, defining the needed intervention of the human operator in robot control, introduced in [47], helps in defining the tasks and their actors. In other end of the spectrum, where no intervention is required from the human, the robot control will be fully autonomy. Level of Intervention and level of Autonomy sums up to 100%. The area between 0% and 100% intervention presents a shared control space; some of the control is done by human and some by the robot [47]. On the other hand, the level of autonomy can be thought as autonomous capabilities of the robot; meaning how well it senses the surroundings and acts and plans after it [48]. In the case of Human-Robot Collaboration, such capabilities enhance the limits of the complexity in collaborative work.

The collaboration can include teaching of the robot. In a case where the robot needs to be programmed online, there are various collaborative paradigms that can be used, such as lead-through programming, walk-through programming, and programming by demonstration (PbD) [49]. The lead-through programming is done using the Teach Pendant of the robot controller. The Teach Pendant is used to jog the robot to the poses or record motions that are then used in the robot code. The walk-through programming is sometimes referred as lead-through programming. In this Thesis, the distinction done in [49] is adapted to get unified explanations. Walk-through programming means that the human will drag the robot by hand, leading the robot to the next pose or also saving the trajectory to get to this pose [50]. Walk-through of the robot requires alternative control strategies. The force of the human dragging the robot needs to be compensated by moving the robot in the direction the force is applied, and the robot needs to be in a state of “floating”. Such compliance robot control can be achieved with impedance, admittance, or stiffness control [51]. Lastly, the PbD explains a setup where the human can program the robot by demonstrating where and how the robot should move. This can be achieved for example with sensors located on the human body, for example using arm, wrist, and head band in [52] and teaching the robot how to pick up objects by providing demonstration with the tracked human hands.

### **2.3 Natural human interaction methods**

Natural communication between the human and the robot implies that the used communication methods should be intuitive for human to use, ultimately like what is present in human-human communication. Such interaction methods are speech, touch, gaze, and body gestures.

When humans communicate with each other, the exchanged message is mostly consisted of body language (55%). The tone of the human voice provides takes 38% and the spoken words only 7% [53]. When deciding what natural interaction method to choose for the interactions with this knowledge, the body gestures would be a fine place to start. Furthermore, humans interact with their surrounding world mostly with their hands [54], and therefore presents as apparent choice as the basis of the interaction.

Speech can be used to give instructions for robot through vocabulary [55]. The speech as a modality requires a speech recognition system. The problem was solved earlier with Hidden Markov Models [56], but recent state-of-the-art systems are implemented with Deep Neural Networks [57]. Speech can be used for example execute predefined robot tasks, by providing required keywords as was done in [58].

The robot can feel a touch when the robot body is equipped with tactile sensors [59] or proximity sensors [60]. Similarly, tactile sensors are needed on gripper surfaces which touches it's environment for the robot to feel the touch [61]. Such sensors can be used as safety measure or can be utilized in systems where the robot needs to gently grip a part and adjust the grip force accordingly.

The eye-gaze usually reveals what the human is focused on given time. This information can be read with vision sensor and used for example in HRI to instruct the robot on what part to pick next [62].

The hand and arm gestures can be interpreted with two types of devices, vision-based devices, and wearable sensors. Furthermore, the wearable sensors can be divided in three categories by the sensors they use, encoders, bending sensors and Inertial Measurement Units (IMU). In [63] authors proposed a benchmark for motion capture, which is presented in Table 2.

Table 2. *Motion capture method benchmark [63]*

<b>Method</b>	<b>Vision</b>	<b>Encoder</b>	<b>Bending sensor</b>	<b>IMU</b>
Accuracy	High	Very high	Low	High
Repeatability	High	Very high	Low	Medium
Wearability	High	Poor	High	High
Portability	Poor	Medium	High	High
Cost	High	Medium	Low	Low

The benchmark works well as indicator of feasibility of the sensors for the applications that requires motion capture. It also gives some insight how the different types of sensors would work with static gestures.

The human arm can be used to communicate with a robot using wrist or armbands [64]. The motion and/or static gestures of human hand can be captured using a smart glove using one of the before mentioned sensor types. Bending sensors in smart glove provides only one value and degrees of freedom (DoF) per finger [65] or additionally two/three if the bending sensors are placed on the joints [66]. Optical encoders have multiple values per finger [67], one for each joint, and therefore provides three DoF per finger. IMU sensors can be used to retrieve the actual pose of each joint of the finger [68], and therefore presents accurate state of the fingers. However, IMU sensors require

more computational power for acquiring the motion and gestures, since more DoF is present. With smart gloves, the drawbacks come from wearability [69] and in many cases difficult calibration [70].

Vision-based approach covers different types of external sensors, such as 2D cameras, stereo vision, and depth cameras, and can be used to recognize human motion and gestures. On top of the feed of the camera, the system needs to be coupled with Machine-Learning (ML) or Artificial Intelligence (AI) for it to be able to spot and recognize gestures [71]. The ML/AI solutions relies strongly on the fact, that there is computational power available [72], which might cause problems in some solutions. Vision-based gesture recognition offers precise recognitions of the gestures by not relying on accuracy or repeatability of wearable sensors. The external sensors offer more input method possibilities in the means of adding the arm or head as input to the pool of possible gestures without adding sensors on the human body. However, the system has some drawbacks in the sense of portability and cost [63], sensibility to occlusion and complex background [73] and the fact that the gestures need to be done facing the sensor.

## 2.4 Bin-Picking

Bin-Picking is seen everywhere, from warehouses to manufacturing processes, yet is a task mostly done by humans [5]. In these countless applications automating the Bin-Picking processes would increase the productivity and has countless applications. For human the Bin-Picking is easy task, but for the robot there are advanced algorithms needed to solve the problem [8]. This robotic problem is called Bin-Picking problem.

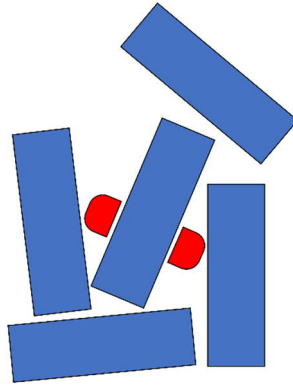
Bin-Picking problem has been researched for decades, but still general solution has not been found for the problem. The problem can be divided to two sub-problems, pose estimation and collision avoidance [7]. For the pose estimation three poses needs to be solved:

- the object pose – object pose in world frame,
- the gripper pose – gripper pose in world frame,
- and the grasp pose – object pose in gripper frame.

Since the poses create a transformation chain, when two of the poses are known, the third one can be computed.

Collision avoidance refers to the need of the poses of the gripper to be analysed and ensure that they will not collide with the environment. The environment here refers to

primarily to other parts in the bin and the bin itself. The Figure 5 illustrates how the red gripper fingers needs to find a place around the blue piece to be able to grasp it.



**Figure 5.** *Cross section of finding grasp position without colliding to other parts*

When creating a Bin-Picking solution, it is important to know if there is any structure for the parts in the bin. The parts can be in structured, semi-structured or random poses [74]. Structured parts would always be in a same pose. Semi-structured are almost structured, for example the parts can have constraints on the angle, or they don't have occlusion. Random Bin-Picking is the hardest Bin-Picking problem since all the parts are randomly located in the bin without any structure.

Gripper fingers have major influence on what kind of parts the Bin-Picking system can pick. State-of-the-art mechanisms are either two finger grippers, suction gripper or combination of these two [75]. Examples of such grippers are presented in Figure 6.



**Figure 6.** *On robot two finger gripper [76], Robotiq vacuum gripper [77] and ABB smart gripper [78] with vacuum and two fingers combined*

There are multiple different approaches to locate objects in a bin. Random Sample Matching Algorithm (RANSAM) can be applied to pick single types of objects [8] or arbitrary objects [9] from a bin. As the algorithm states to be matching algorithm, this means that the 3D models of such parts are known. This is a premise in many algorithms regarding the Bin-Picking since in many cases the pose where the part is taken after the



Bin-Picking by the robot needs to be always the same. The grasp pose can be determined with Key Grasp Frames introduced in [9] using the 3D model of the gripper and object pose to determine the grasp position.

After the location of the part is solved, the grasp pose needs to be found and evaluated. The state-of-the-art method for collision avoidance of the grasp pose usually uses point cloud based solution, such as RAPID-algorithm based on OBBTree [79].

For the Bin-Picking application to be able to handle unknown objects, many of the solutions makes simplifying assumptions. For example in [80] authors assumes that the objects are planar, and in [81] the objects are laid on flat surface and are always picked up from above the part. In [82] the authors presents algorithm without the assumptions and presents of high quality results for autonomous Bin-Picking with unknown data.

### **2.4.1 Point Cloud processing**

When creating the algorithm that finds the required grasp pose for picking the part, that has a point cloud as input, there is a need for clustering/segmentation of the parts. The segments can be compared to the 3D model of the object, or further processed for finding the grasp pose. For example, in a case where the robot gripper uses a vacuum grip, the Random Sampling Algorithm Consensus (RANSAC) [83] algorithm can be used to segment planes from the point cloud and with these planes calculate a grasp-pose, such as in [84].

A fast algorithm, yet not the most state-of-the-art, for segmenting point clouds is DBSCAN algorithm [85]. This algorithm is based on density-based notion of clusters and finds clusters of arbitrary shape. In [86] authors propose an efficient method for point cloud with connected components. The more the parts are on top of each other and are occluded, the more complex segmentation algorithm is needed.

Today, when ML and AI applications are regularly used in new implementations, such algorithms can provide a powerful and fast methods for segmentation and pose estimation. Good example of this is the PPR-Net [87] which utilizes Point-Net++ [88] to segment and estimate the poses of the objects.

## 3. DESIGN

Bin-picking problem can be characterized as hard robotic problem, and one large reason is the lack of universal solution – the solutions are not flexible enough to be universal. Majority of Bin-Picking solutions relies on the fact that 3D models of the parts to be picked exist. However, with certain restrictions it is possible pick parts without knowing the 3D model before-hand. When solving the problem using an HRC, it is beneficial to exploit the natural human interactions as communication method. Modularity and flexibility of the system should be considered by creating components that have APIs for unified middleware.

First in Sec. 3.1 the hardware components of the solution are identified, and the deployment type of the software components is introduced. The Sec. 3.2 provides information about how the design paradigms of HRI/HRC are applied to the solution, and Sec. 3.3 shows the high-level software architecture and the communication methods used between the later proposed components.

### 3.1 Components

There are three hardware components used in this Thesis: CaptoGlove smart glove, ABB IRB14000 YuMi dual-arm collaborative robot and Zivid 3D camera. These components and chosen software related design choices are introduced in following paragraphs. The physical setup is assembled in FAST-lab. FAST is an acronym of *Future Automation Systems and Technologies*, and does research in automation, CPS, AI, robotics, and industrial informatics [89].

CaptoGlove falls in a cheaper spectrum of commercially available smart gloves [90]. It has two types of sensors, bending sensors and pressure sensors, and are located as illustrated in Figure 7. The glove has two textile layers, where in between the bending sensors and pressure sensors are located in. The sensors are not attached to the glove, which helps in a situation where the sensor needs to be changed if malfunctioned, or if the textile of the glove needs to be washed. On the other hand, since the sensors are not rigidly attached to the glove, the sensors might move when putting the glove on or taking it off, which might affect the reading of the sensor values.



**Figure 7.** Locations of the sensors, on left palm side down picture the bending sensors and pressure sensors in right palm side up picture are presented in blue color and the transmitter is presented in orange color

Due the fact that the Thesis tries to find a solution to Bin-Picking problem through Human-Robot Collaboration, the robot should support collaborative procedures. For this task, the collaborative robot IRB14000 ABB YuMi (Figure 8) was chosen. YuMi robot has two hands, where there is 0.5kg load capacity in each [91]. YuMi has also Smart Gripper attached to each end-effector, which both weighs 262g, setting the maximum load capacity to 238g [92]. In this Thesis the suction grippers are not used, since the robot cell did not have air pressure installed. YuMi is created for the collaborative tasks, and therefore has safety aspects built-in, such as soft paddings in the arms (in dark grey colour in the picture) and torque and force sensors in the joints.



**Figure 8.** ABB Yumi Robot [91]

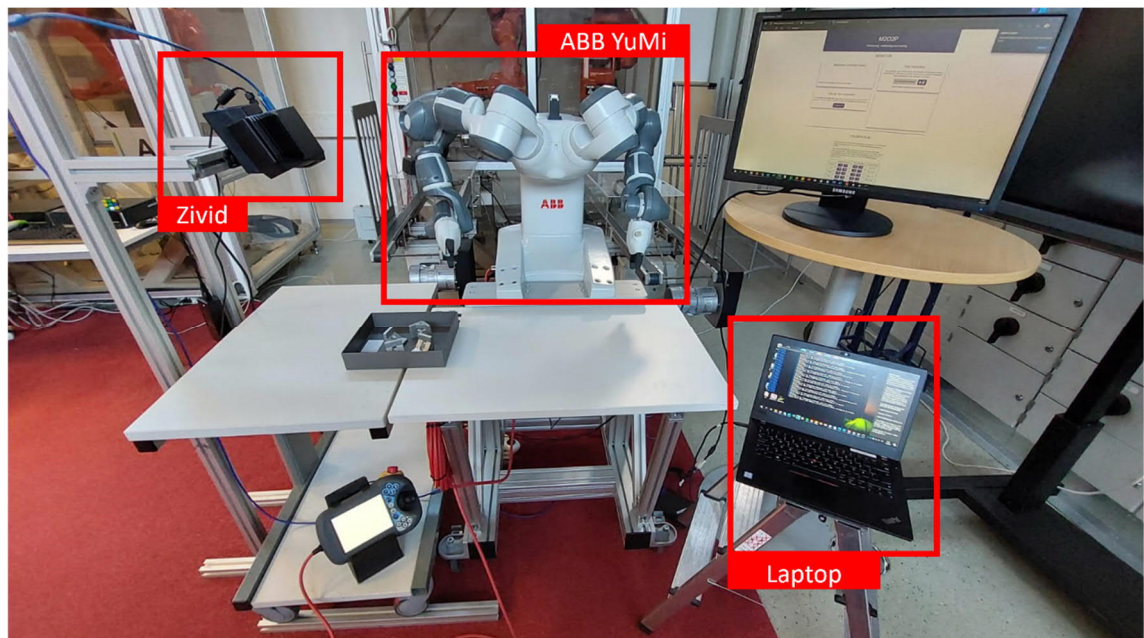
The last hardware component, i.e., 3D camera, was chosen to be Zivid One+ S. Zivid One+ is a 3D camera that is designed for Industrial robot cells [93]. The camera will output a 1920 x 1200 (2.3 megapixels) RGB-colored point cloud with 25  $\mu$ m point preci-

sion. Such density and precision provide optimal setup for accurate point cloud processing and finding the required poses for Bin-Picking. The camera is presented in Figure 9. The Zivid camera uses eye-safe white structured light to capture the 3D data.



**Figure 9.** Zivid One+ 3D camera [94]

These hardware components are displayed in Figure 10. To provide a modularly available 3D camera for robot cells, there was a movable stand assembled from same construction pillars used in other robot cells. The Zivid One+ S has working range of 300mm-1000mm [93], which was considered when assembling the stand for it. This stand is located on the left side, the robot in the middle, and the laptop used to run the software components on the right side. The table in front of the YuMi acts as the collaborative workspace and the movable table on the left is reserved for the bin as an extension of the robot workspace.



**Figure 10.** The cell collaborative robotic cell layout

For deployment of the software components, OS-level virtualization using Docker [95] containerization was used. Containerized applications are system agnostic and offers a

solution that is easy to deploy and configure. Configuration of the Docker images is done using Docker volumes [96]. In this Thesis, the volumes are used to mount configuration files inside the Docker image, for example giving runtime variables for the software components. Another useful function of volumes is that the file is shared between the Docker container and the host, which means that changes done inside the Docker container will hold if the system is shut down and put on again.

### 3.2 Solution Design

In robotic applications that follows the model of *Industry 4.0*, flexibility and modularity are the key attributes that are required from such system. The Use-Case was created and solved with these aspects in mind.

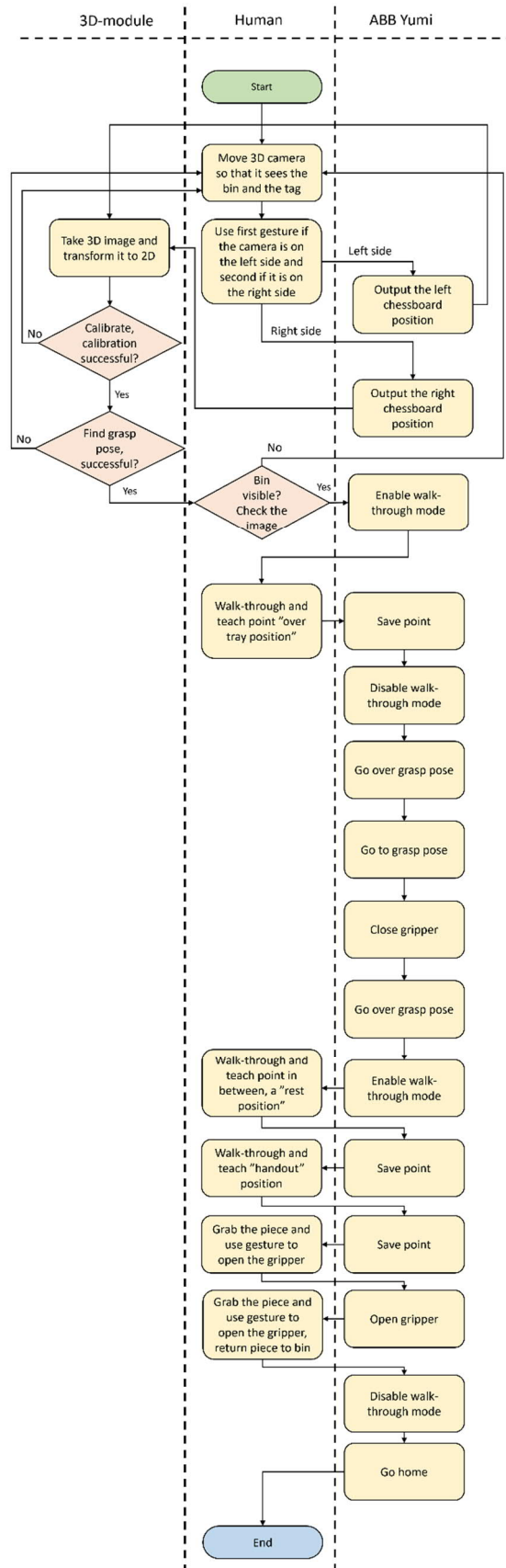
As introduced in literature review, there are five attributes that can be chosen for HRI system. For the solution of this Thesis the attributes of HRI were selected as presented in following table (Table 3).

Table 3. *HRI attributes selected for the solution*

Attribute	Explanation
Level of Autonomy (LOA)	For the whole solution the human intervention – robot autonomy ratio is 50-50. The solution utilizes the flexibility of the human operator who will teach the robot needed information for three robot poses. The taught system will automatically pick up the parts and hand them to the operator.
Nature of information exchange	The human will interact with the robot with gestures and using the UI of M2O2P component. The robot will not communicate with human any other way than executing tasks.
Structure of the team	The team is composed of human operator and two-handed ABB YuMi robot.
Adaptation, learning and training	M2O2P application will teach the human to use gestures. Robot will learn from walk-through the required poses.
Shape of the task	The process will have endless possibilities for example for different modalities. Modular components can be used in other Use-Cases that uses the FIWARE as middleware

The human and the robot will share a workspace, workpiece, and the goal, leading the HRC level to be collaboration. The shared environment implies that the safety measures for ensuring safe collaboration needs to be considered. The YuMi collaborative robot will ensure the safety through the safety paddings installed to the arms and with safety through control, using ABB Collision Detection [42]. Furthermore, the speed of the robot movement will be adjusted so that the psychological safety will be ensured.

The following process flowchart presented in Figure 11, which describes a process where human operator teaches the robot to do Bin-Picking operations and hand the picked parts for the human operator. The flowchart has three lanes for different actors, the robot, the human operator, and 3D-module. The orchestrator will assign the tasks for the actors and proceeds in the process execution once the task is completed. Such task-based approach for the process flow is highly modular, but also expects the modularity from individual components in the form of available functions.

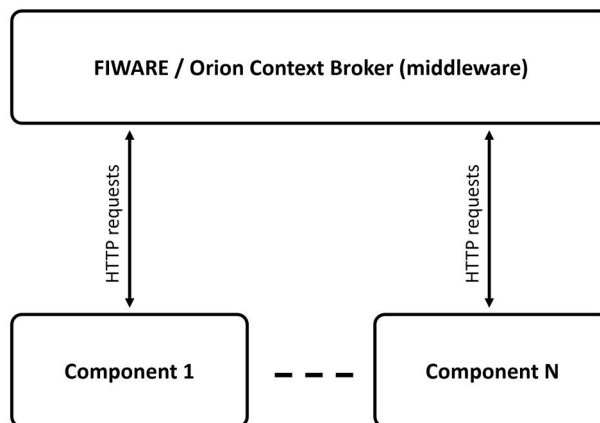


**Figure 11.** Process flow of the teaching process

In short, the process will calibrate the camera location, find grasp pose and with the help of human teach the required poses for the robot. After the process, the system is capable to do Bin-Picking operations by picking an object from the bin and handing it out to the human operator.

### 3.3 Architecture and Communication

To comply with the idea of Smart Factory, the components created for the solution are all context-aware and connected to same middleware, FIWARE. FIWARE offers a platform for IoT devices that can update and consume context information [97]. The FIWARE setup consists of two actors, Mongo Database (MongoDB) and Orion Context Broker (OCB). OCB is used to manage context information including creation, updates, subscriptions, and registrations [98] through HTTP requests (GET, PUT, POST, DELETE). The MongoDB [99] is used to save this context information, that is created by OCB. The FIWARE connection of developed components can be seen from Figure 12, where all the components have an interface with FIWARE through HTTP-requests.



**Figure 12.** General architecture of the system and components within it, and their connections to FIWARE.

For the components to communicate with each other seemingly, common language between the actors is required. For this task the data models from SHOP4CF [100] project were adopted in the use of this Thesis. The communication through FIWARE is established by using three types of entities, Task Entity, Resource entity and Subscription entity. The first two data models are execution data models, which means that they change statuses dynamically and provides information about what is occurring in the shop floor. The entities exchanged through FIWARE are in JSON format. Since the solution will use NGSI-LD [101] format, every entity holds a *@context* field for context information. In this solution, every entity will hold the smart data models context, and the



special context information made for SHOP4CF project, which can be seen from example entities presented later in this chapter.

Task entity has information about the task, who executes it, what happens in the task and where it happens [102]. The Task entities are the main communication method between components in the solution. The status of the Task entity has three states, *pending* when it is created, *inProgress* when the involved component has accepted it, and *completed* when the component has completed it. Example of the Task entity is presented in Figure 13.

```
{
  "id": "urn:ngsi-ld:Task:fastlab:example",
  "type": "Task",
  "isDefinedBy": {
    "type": "Relationship",
    "object": "urn:ngsi-ld:TaskDefinition:fastlab:example1"
  },
  "involves": {
    "type": "Property",
    "value": [
      {
        "type": "Relationship",
        "object": "urn:ngsi-ld:Device:fastlab:device-1"
      }
    ]
  },
  "workParameters": {
    "type": "Property",
    "value": {
      "command_id": "",
      "taskDescription": ""
    }
  },
  "status": {
    "value": "pending",
    "type": "Property",
    "observedAt": "2020-12-01T11:23:19Z"
  },
  "outputParameters": {
    "type": "Property",
    "value": {
      "continuation": "",
      "output": {}
    }
  },
  "observedAt": "2020-12-01T11:23:19Z"
},
"@context": [
  "https://raw.githubusercontent.com/shop4cf/data-models/master/docs/shop4cfcontext.jsonld",
  "https://smartdatamodels.org/context.jsonld"
]
}
```

**Figure 13.** Example of the Task entity

The most important attributes of the Task entity are the *involves* attribute that holds the information on which actors are involved in the task, *workParameter* that holds information about the task such as the identifier of the command or the human-readable task

description, *status* that is used for task status updates as explained earlier, and *output-Parameters* that will hold the output information from the involved component.

The Resource entity holds status of a resource, which can be person, asset, material, or device. In the solution, for every component there is Device entity [103] created. In this solution person availability was not considered. The person related tasks are completed using the CaptoGlove, and there is no materials or assets used, so they are not introduced. The example of the Device entity is illustrated in Figure 14.

```
{
  "id": "urn:ngsi-ld:Device:fastlab:device-1",
  "type": "Device",
  "controlledProperty": {
    "type": "Property",
    "value": ["commandId"]
  },
  "deviceState": {
    "type": "Property",
    "value": "shutdown"
  },
  "commandId": {
    "type": "Property",
    "value": 0,
    "observedAt": "2020-12-01T11:23:19Z"
  },
  "@context": [
    "https://smartdatamodels.org/context.jsonld",
    "https://raw.githubusercontent.com/shop4cf/data-models/master/docs/shop4cfcontext.jsonld"
  ]
}
```

**Figure 14.** Example of the Device entity

The device entity holds a *controlledProperty* that describes properties that are updated by the component, and the corresponding properties are found from the Device entity itself. The *deviceState* describes the status of the component, which are either *ok* or *shutdown* and provides the availability status of the component.

The subscription entities are created by the components and the subscription notifications are triggered by the changes done in the subscribed entity types. Example of subscription entity is provided in Figure 15.

```

{
  "id": "urn:ngsi-ld:Subscription:subscription-task",
  "description": "Send notification to this device",
  "type": "Subscription",
  "entities": [
    { "type": "Task", "idPattern": ".*" }
  ],
  "q": "involves.object==urn:ngsi-ld:Device:fastlab:device-1",
  "notification": {
    "attributes": [
      "id",
      "status",
      "workParameters"
    ],
    "format": "keyValues",
    "endpoint": {
      "uri": "http://device:54200",
      "accept": "application/json"
    }
  },
  "@context": [
    "https://smartdatamodels.org/context.jsonld",
    "https://raw.githubusercontent.com/shop4cf/data-models/master/docs/shop4cfcontext.jsonld"
  ]
}

```

**Figure 15.** Example of the Subscription entity

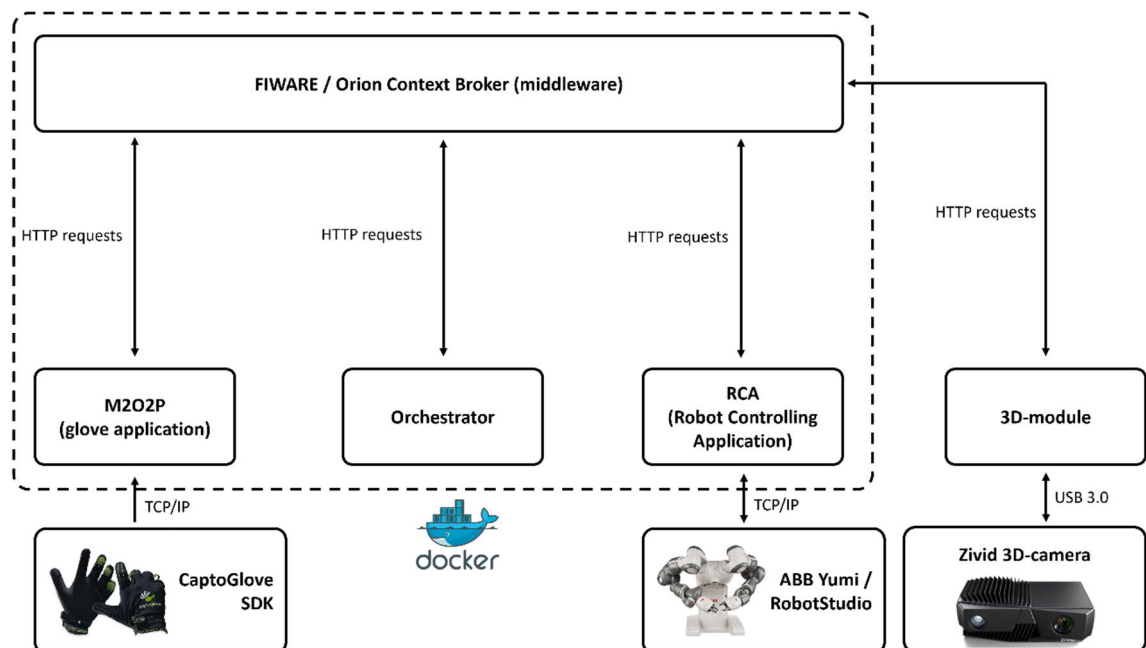
The subscription entity will subscribe to all Task entities with all identifiers and creates notification to the provided endpoint if the *object* attribute of *involves* matches the device identifier of the subscriber. The notification will hold the identifier, status, and work parameters and they will be presented as key-value pairs.

## 4. PROPOSAL

To answer to the requirements designed for the system, this chapter presents the proposed method used in the solution. Sec. 4.1. presents the architectural and communication information created for the whole system. Sec. 4.2. will focus on the proposed component for handling the human-robot interface, Multimodal Offline and Online Programming (M2O2P) solution. Sec. 4.3. presents Orchestrator Application (OA) that handles process management, Sec. 4.4. focuses on the robot control and Robot Controlling Application (RCA), and at last the Sec. 4.5. presents the 3D-module component and provides information about how the 3D data provided by the Zivid camera is processed.

### 4.1 Architecture and Communication

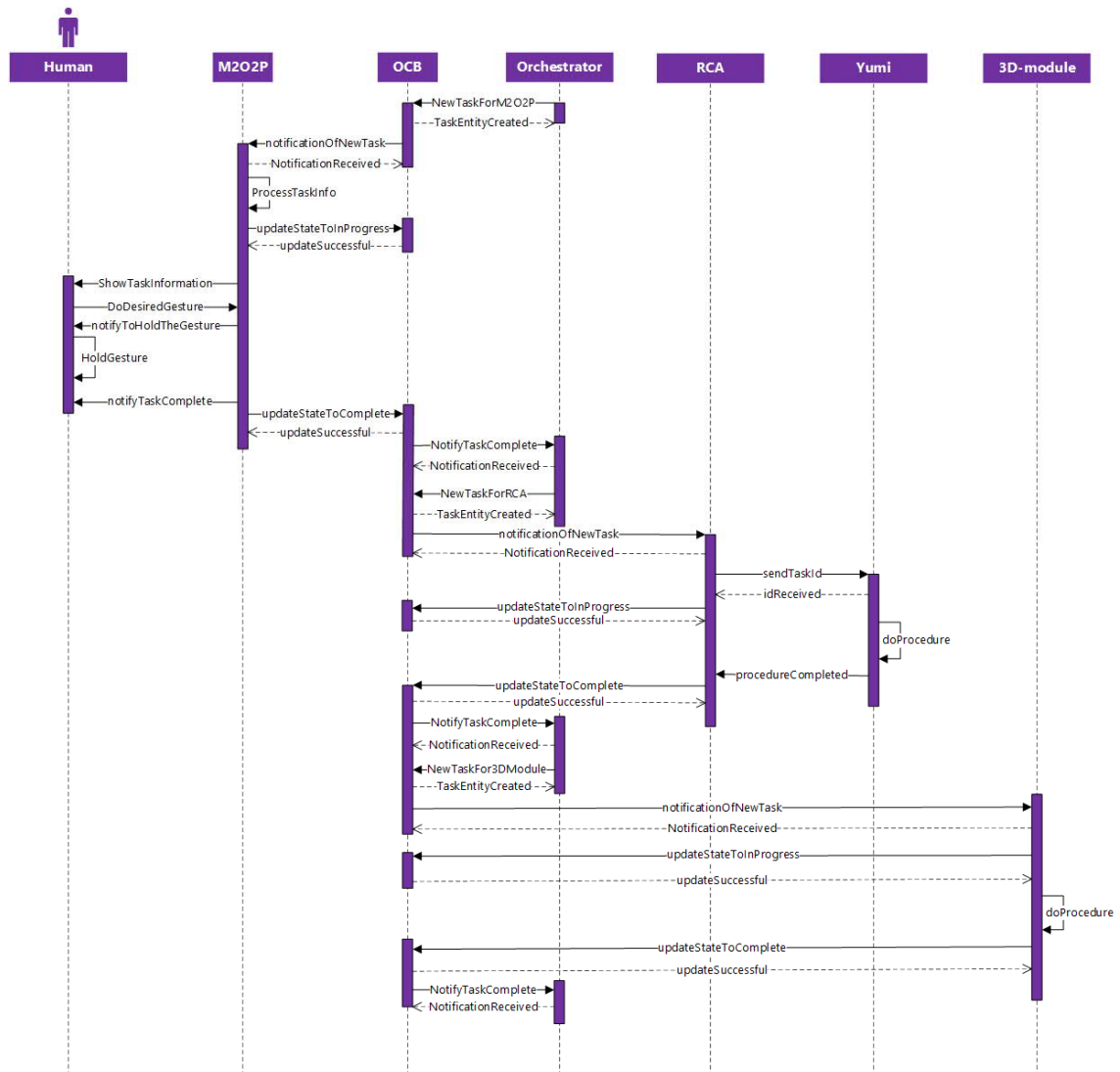
Architecture of the system was designed in Smart Factory setup with the help of FIWARE. FIWARE exposes the exchanged context information to other FIWARE compatible components and therefore makes the solution scalable. The architecture and the communication methods between the actors are illustrated in Figure 16.



**Figure 16.** Architecture diagram of the solution

The software components are containerized in Docker (except 3D-module), and communication with the hardware is established with TCP/IP communication or with USB 3.0.

Following sequence diagram (Figure 17) was created to illustrate the communication between the actors within the solution in a scenario where each software component gets a task from the orchestrator one after the other.



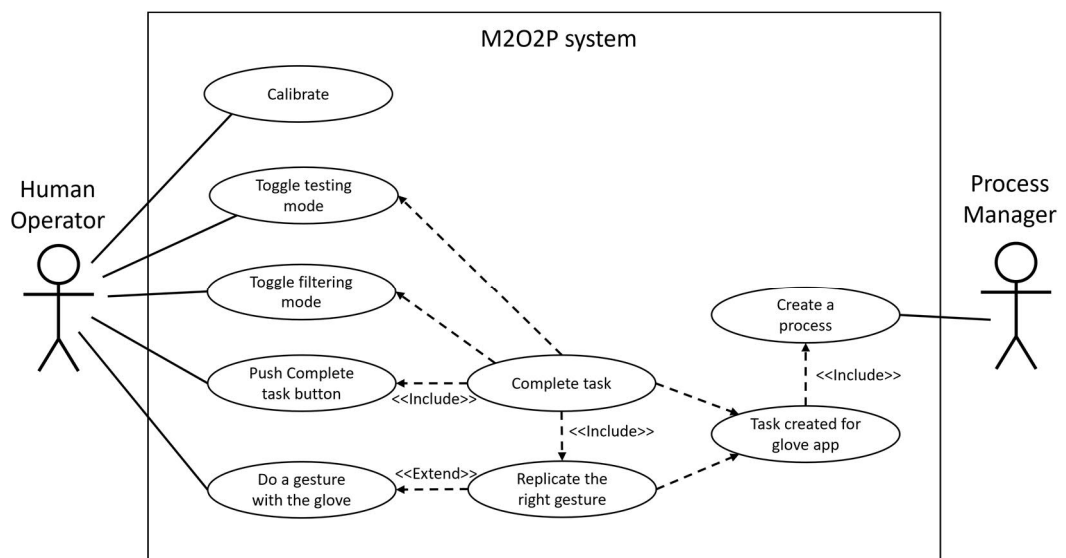
**Figure 17.** Communication between the components

Orchestrator creates Task entity that has *bending* status. All actors have subscribed to Task entities that involves their identifiers, and therefore gets a notification from the FI-WARE. When M2O2P has received the task, it will process the notification and after ensuring that needed information is provided in the Task entity, it updates the state of the Task entity to *inProgress*. Then M2O2P updates the User Interface to show the information to the human operator. Human operator does the desired gesture and holds it, and after held for 1.5 seconds in total M2O2P updates the status of the Task entity to completed. Similar actions can be seen when the orchestrator assigns task for RCA and 3D-module.

## 4.2 M2O2P

Multimodal Offline and Online Programming (M2O2P) solution was developed in the SHOP4CF project and was used as human-robot interface in this Thesis. Since the developed component was created for HRI, it was beneficial to use the Robot Operating System (ROS) [104] as middleware between the modules of the component and can be used for direct communication. ROS provides publish-subscribe communication between ROS nodes, using ROS topics [105]. M2O2P uses ROS2 Foxy Fitzroy [106] for the communication within the modules and provides the output information of the application additionally in ROS1 topics. Docker containers that are deployed together natively shares the ROS topics with each other, and therefore presents as an excellent middleware option.

In the process, M2O2P is the only component that is extensively interacted by the human operator, and therefore an UML Use-Case diagram helps in understanding how the component works. In the design phase of the component the following Use-Case diagram was created to illustrate the interaction (Figure 18).



**Figure 18.** Use-Case diagram of human operator interacting with M2O2P component

The human operator has five options for interaction namely, do a hand gesture using the glove, push the complete button, toggle filtering mode, toggle testing mode and calibrate the application. These modes are explained later in the Thesis. The process manager can create a process, that has a task that involves M2O2P. When there is a task, the task can be completed using the complete task -button or doing the right gesture. However, both are reliant of the filtering mode to be on and testing mode to be off, meaning that task cannot be completed if one of these are not true.

The used smart glove, CaptoGlove has two types of sensors, bending sensor and pressure sensors. In early design and development, the pressure sensors were set to be used with the bending sensors. This however came out to be difficult since the pressure sensors weren't working reliably when the fingers were bent. The pressure sensors would have given unique gesture possibilities, by on top of the specific states for bending sensors, some fingers would have been pushing each other. There were 21 different gestures chosen to be used in the application, and their corresponding states are shown in Table 4. There are three states for each finger, straightened, something in between and bent. The state 2 is bent and 0 is straight.

Table 4. *Gestures and their corresponding bending states*

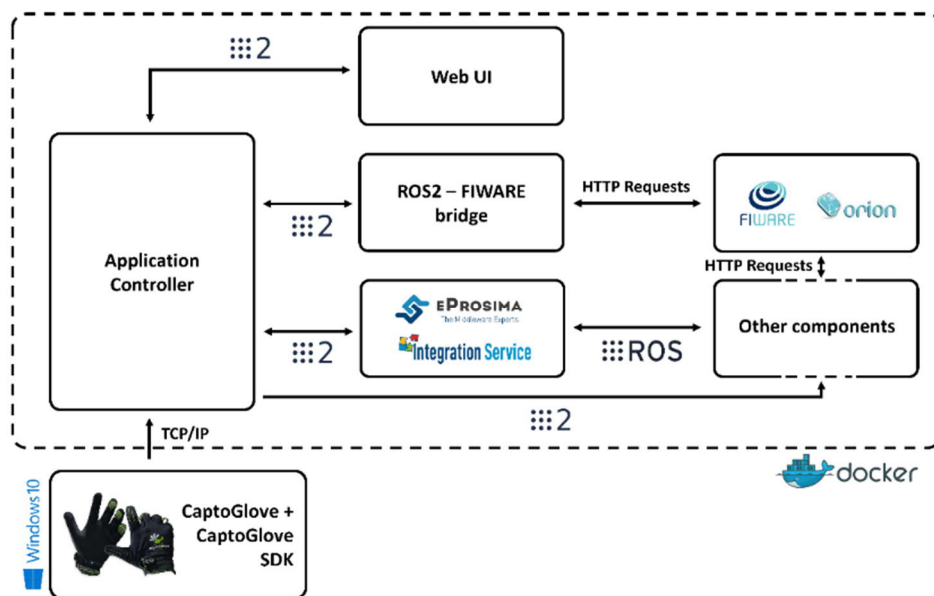
<b>Gesture</b>	<b>Thumb</b>	<b>Index</b>	<b>Middle</b>	<b>Ring</b>	<b>Pinky</b>
Horns	2	0	2	2	0
Pinky straight	2	2	2	2	0
Point with index	2	0	2	2	2
Thumb and index straight	0	0	2	2	2
Thumb and pinky straight	0	2	2	2	0
Middle straight	2	2	0	2	2
Middle and pinky straight	2	2	0	2	0
Index and middle straight	2	0	0	2	2
Ring and pinky straight	2	2	2	0	0
Index, ring and pinky straight	2	0	2	0	0
Middle, ring and pinky straight	2	2	0	0	0
Index, middle and pinky straight	2	0	0	2	0
Index, middle and ring straight	2	0	0	0	2
Middle and ring straight	2	2	0	0	2
Index and ring straight	2	0	2	0	2
Ring straight	2	2	2	0	2
Thumb, index and pinky straight	0	0	2	2	0
Thumb, index, middle and pinky straight	0	0	0	0	2
Thumb, middle and pinky straight	0	2	0	2	0
Thumb, index and middle straight	0	0	0	2	2
Thumbs up	0	2	2	2	2

In a development phase, the application was designed to consist of five modules. The modules and their uses in the implementation are explained in a Table 5.

Table 5. Introduction of the modules used in M2O2P component

Module, abbreviation	The purpose in the solution
CaptoGlove SDK (SDK)	Read the raw sensor data from the smart glove and send it to Application Controller
Application Controller (AC)	Process sensor data to gestures and ultimately to commands, communicate with Web UI and ROS2-FIWARE bridge
Web User Interface	Graphical User Interface provided for the user to enable communication with the application
ROS2-FIWARE bridge (bridge)	Configurable bridge between ROS2 and FIWARE using NGSI-LD information model
Integration Service	Provides bridge between ROS2 and ROS1

The components are further explained in their corresponding sections in Implementation and Testing -chapter. The software architecture of M2O2P including all the components is presented in Figure 19.



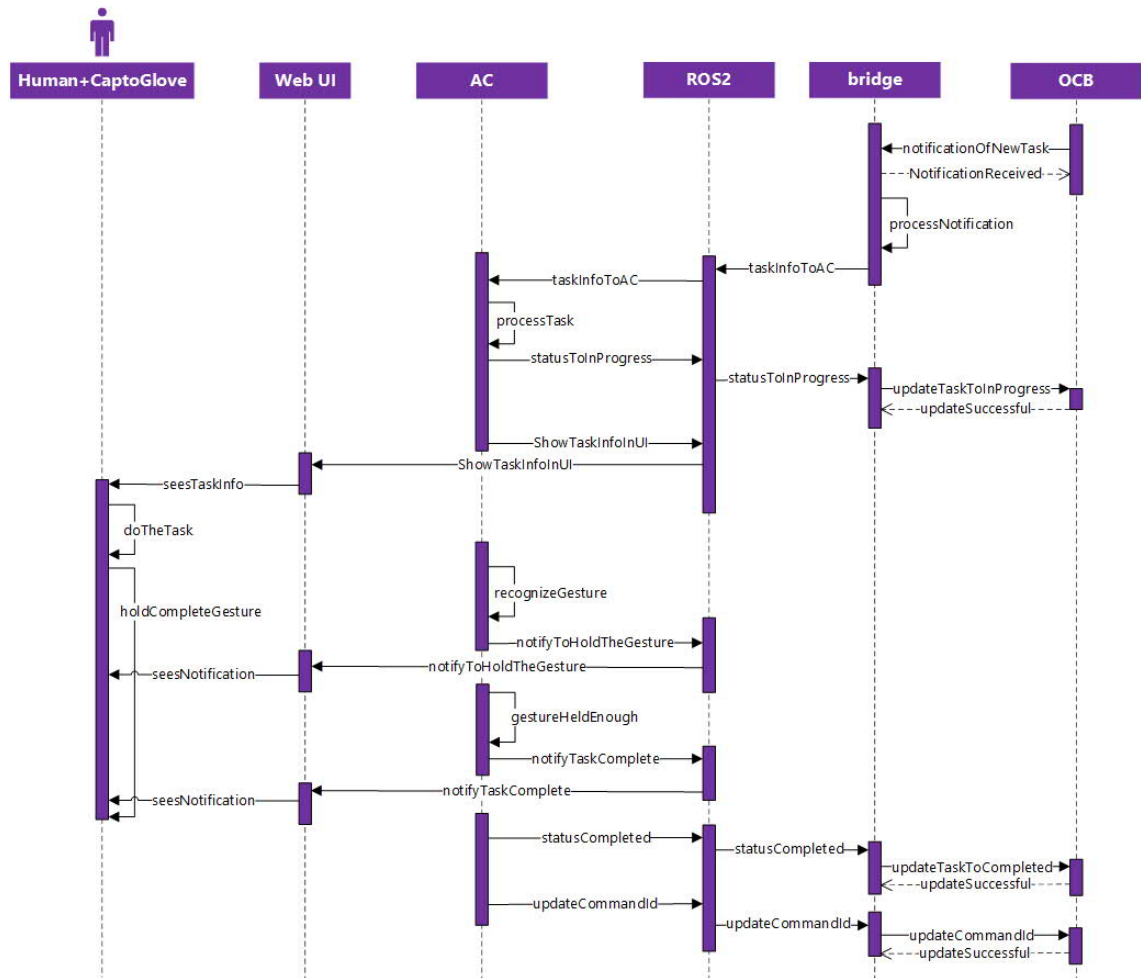
**Figure 19.** Software Architecture of M2O2P

The Application Controller (AC) receives the sensor data over TCP/IP connection from CaptoGlove SDK and communicates with other modules using ROS2 topics. The ROS2-FIWARE bridge handles the communication between the component with ROS2 and FIWARE with HTTP-requests.

The communication between the components was already presented in Sec. 4.1, but since M2O2P consists of multiple modules, sequence diagram was created to explain



the communication inside the component. The sequence diagram is presented in Figure 20.



**Figure 20.** Sequence diagram of M2O2P receiving and completing a task

When there is task created for M2O2P, the bridge gets the notification, processes the notification, and sends the relevant task information to AC. AC will process it and invoke the bridge to update the task to *inProgress* state. Then the information is shown in UI and seen by the human operator wearing the CaptoGlove. The human operator does the task, if there is one on top of doing the gesture and starts to hold the gesture. This is recognized by AC which then notifies through UI that the user should hold the gesture. Furthermore, when the gesture is held long enough to count it as completed task, it will be again notified to the user through the UI. Status of the Task entity and command identifier of the Device entity will be updated with the help of the bridge. The CaptoGlove will send the sensor data to AC for the whole time, and the gestures are read when it is required.

### 4.3 Orchestrator Application

Orchestrator Application (OA) is created to handle task assignment and process flow of the solution. Since the Thesis doesn't focus on process management, the component is mainly hard coded, but still aims to be configurable for further usage in other solutions.

The OA uses Task entities to provide the task information for the consuming component by publishing them to FIWARE. The process is provided for the Orchestrator Application as a list of JSON (JavaScript Object Notation) entities that provides the necessary task information. The approach aims to provide the task list as unified format, and if the OA would be further developed to provide GUI-based process modelling, the interface could use similar format for exchanging the data between the modules. These entities are further explained in the Implementation and Testing -chapter.

OA will handle three types of tasks, static tasks, tasks that expects output from the component and tasks that requires an input to be given for the component. These functionalities are used when assigning tasks, depending on the Task description in the JSON.

There is a need for OA to save variables that other components outputs, which can be invoked later in the same or in other processes. This is handled by saving the variables in Python dictionary as they are output from other components, and additionally saved in text file. This text file is provided for the OA as Docker volume, which means that the changes will be saved even if the Docker container is stopped and started again.

### 4.4 Robot Control

The modularity of the robot control is achieved using static functions in the ABB Robot Controller, that can be invoked through the FIWARE having the Robot Controlling Application (RCA) in between as interface. These static functions are created identical for both of the YuMi robot hands. Robot Controlling Application (RCA) has very similar architecture to M2O2P and ROS2-FIWARE bridge combined. All the tasks are sent to ABB Robot Controller via TCP/IP connection, with own connection for each robot hand.

The solution teaches the robot new poses, which needs to be saved. As this is handled in OA, there is an additional text file created for this purpose where RCA saves the points. The saved points are named as the OA has guided them to be saved. This way OA and RCA both knows the names of the saved points, which therefore can be used later in the process.

## 4.5 3D-Module

To answer the needs of *Industry 4.0*, the robot cells need to be reconfigurable ideally and preferably with automatic methods. When the use of the robot cell will be Bin-Picking, it is beneficial to have the vision hardware available in multiple different robot cells.

To find grasp poses using unknown objects, usually there are some limitations set for the objects that the system can pick, even though this is not the case in state-of-the-art systems. The function created for 3D-module for finding the grasp pose included few limitations on the parts and their placement for the system not to exceed over the main topic of the Thesis, being HRI/HRC. These limitations are follows:

- Parts that are picked must be formed mainly from 90-degree angles
- Parts are picked from above
- Parts are semi-structured, in other words not occluded, but otherwise in random poses

3D-module tackles the modularity requirement by providing camera location calibration function, 3D data acquisition function and function that is used to find the grasp pose. These three functions are presented and explained in the implementation section. The modules use the Open3D [107] Python library for 3D data processing. 3D-module communicates with the Zivid camera through USB 3.0 connection with the provided Zivid Python library.

## 5. IMPLEMENTATION AND TESTING

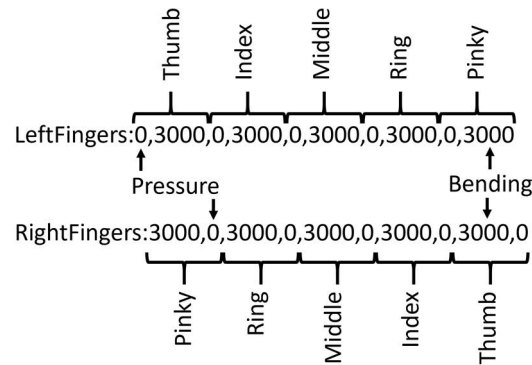
The Use-Case of the Thesis was implemented in the FAST-laboratory in Tampere University. The component for human-robot interface, M2O2P, is presented in Sec. 5.1 There were three other software components created that are presented in this chapter: Orchestration Application (OA) in Sec. 5.2, Robot Controlling Application (RCA) in Sec. 5.3.2 and 3D-module in Sec. 5.4 The implementation of the robot controller is explained in Sec. 5.3.1. The last subchapter Sec. 5.5 presents the testing procedures created for the solution.

### 5.1 M2O2P

This chapter will present all the modules that was developed for M2O2P component. Sec. 5.1.1 provides information of the application that was used to read the sensor data, Sec. 5.1.2 presents the Application Controller created for the component, Sec. 5.1.3 shows the UI of the component, Sec. 5.1.4 the bridge created for communication between the ROS2 and FIWARE, and at last in Sec. 5.1.5 the Integration Service module is presented.

#### 5.1.1 CaptoGlove SDK

CaptoGlove Software Development Kit (SDK) was a C++ project, that was compiled to Windows executable. The basic sensor reading version of the SDK was provided by the CaptoGlove company [108], which was modified to send the sensor data via TCP/IP to the Application Controller. The main function of the SDK is to connect to the smart glove with Bluetooth, read the data from each sensor, and send it as a string to AC. Every other value of the string will hold information of pressure sensor and every other will hold information of bending sensor. Since the devices in both hands are the same, yet the order of the fingers from left to right is different, the sensor values are in reverse order in the left hand. To illustrate the situation, the Figure 21 presents sensor values the case that for all the fingers would be straightened (value 3000), and no pressure is applied to any of the pressure sensors (value 0).



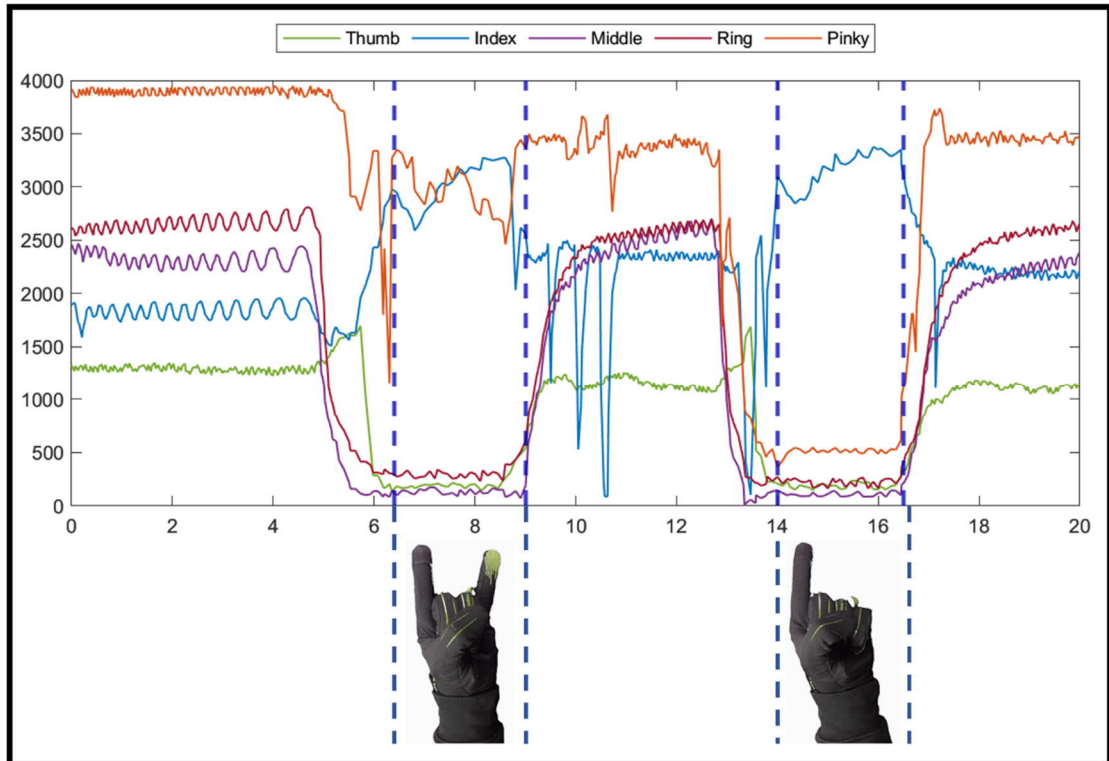
**Figure 21.** The format of the message sent from SDK to AC

The initial plan was to have the SDK also containerized, but since Bluetooth devices were not supported from Windows machine to Linux Docker container, the final decision was to have the SDK as executable on host machine.

### 5.1.2 Application Controller

Application Controller (AC) acts as a brain of M2O2P. AC processes the raw sensor data that comes from CaptoGlove SDK and interprets the sensor values as gestures and finally as commands. Using the ROS2-FIWARE bridge, AC gets the Task information from the Orion Context Broker. Furthermore, AC handles additional options and modes invoked from Web UI such as calibration of the glove, testing mode and ability to affect if the gestures are filtered by the task. These modes are further explained in the Sec. 5.1.3.

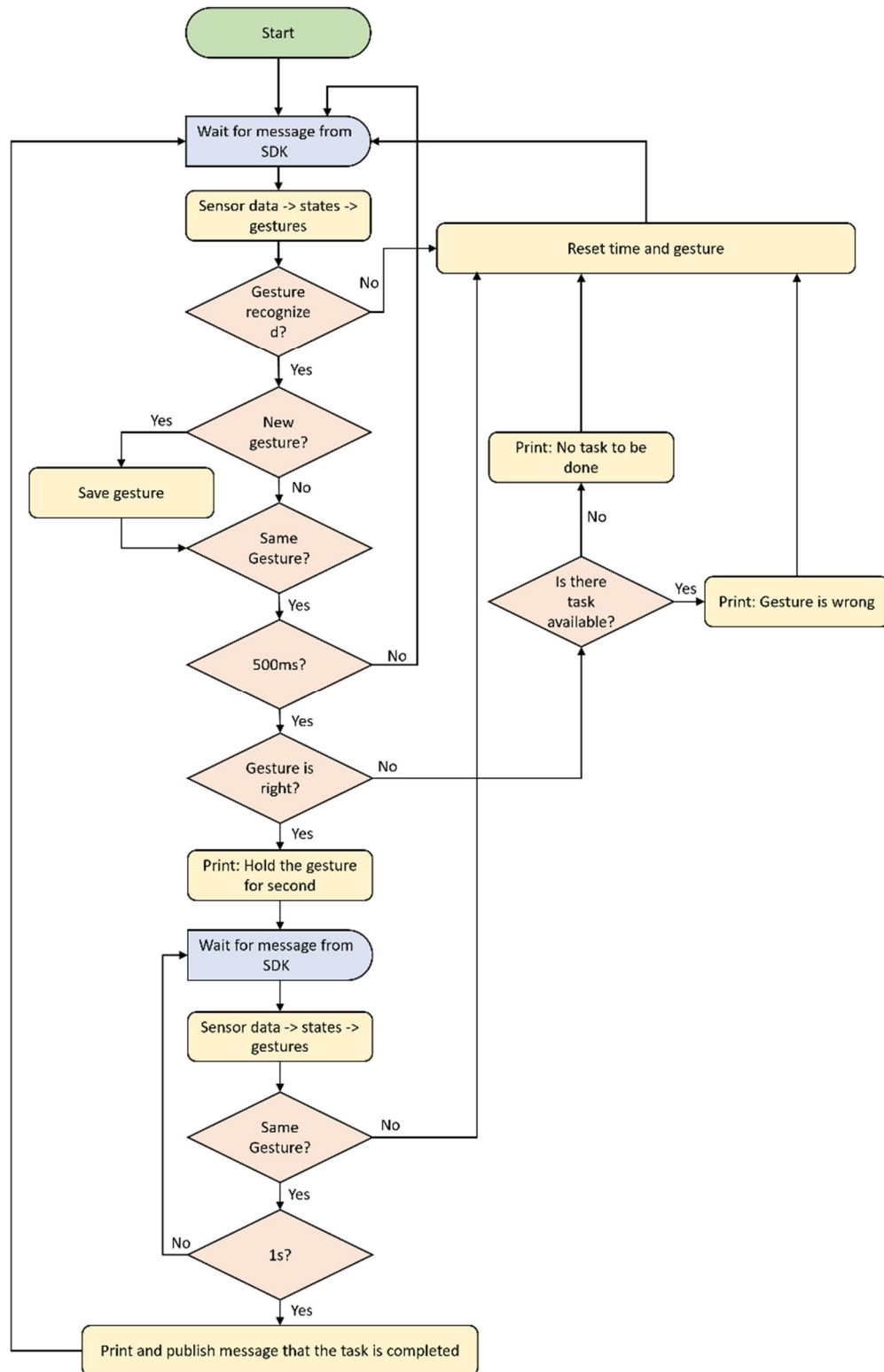
The sensor values for each finger comes to AC as raw sensor values between 0 and ~4000. These values are read as states, that are controlled by the thresholds given for the Application Controller as Docker volume and can be modified using the Web UI. The thresholds are set for each finger individually, since there are a lot of differences in the sensor values between the fingers. In Figure 22 the sensor values are illustrated for 20 seconds. Within this 20 seconds, two gestures were made, between 6.4s and 9s marks the *Horns* was held, and similarly between 14s and 16.5s marks the *Point with index* was held. In between all the fingers were held straight.



**Figure 22.** *Sensor readings of each finger and two gestures done within 20 seconds time window*

There are some disturbances in the sensor values especially with the index finger. These disturbances did not affect the gestures and are short enough to not cause accidental gesture readings.

There are a lot of steps in reading the sensor data and interpreting them as states, gestures and ultimately commands. Therefore, following Figure 23 presents the flowchart of the algorithm.

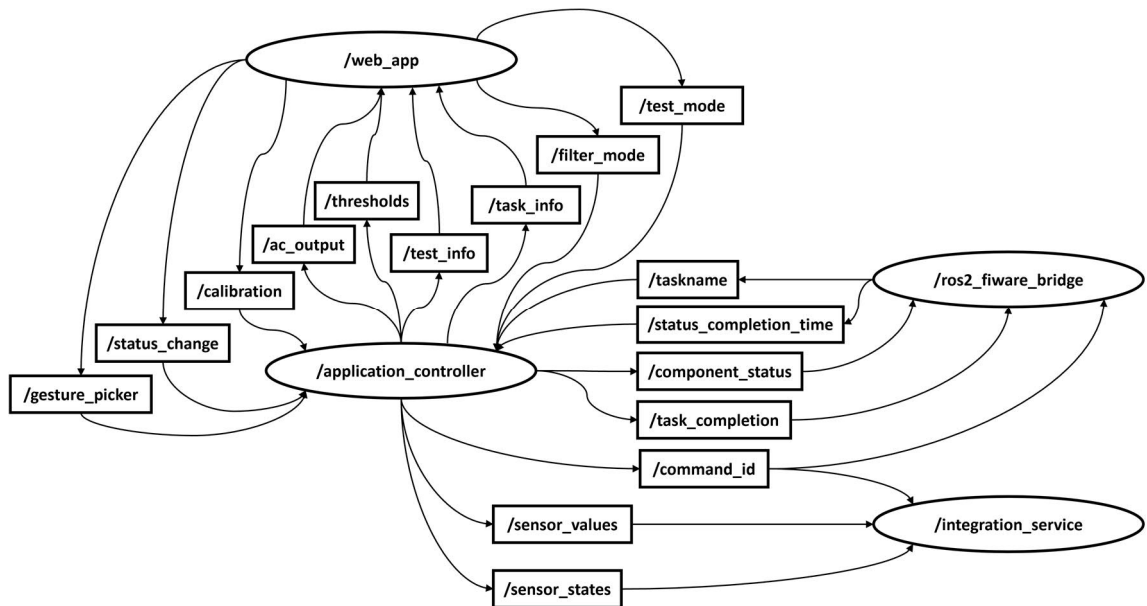


**Figure 23.** Flowchart of AC sensor data processing

The message is received from SDK, and then processed to states and so on to gestures. If the gesture is recognized, AC waits for 0.5s before prompting about the right or wrong gesture. If the right gesture is then held for additional one second, the task is considered

as completed. The gesture needs to be therefore held for 1.5 seconds for the application to read it as successfully done command.

The sensor reading is done in a Python thread to enable background activity and communication with other modules. Application Controller subscribes to seven and publishes to eight ROS2 topics. This information exchange is further explained and illustrated by following communication graph (Figure 24). The graph is created in a style of “rqt\_graph”, which is package ROS provided to visualize the ROS computation graph [109]. The graph describes how the ROS2 nodes, here ellipses, communicate with each other by publishing and subscribing to ROS2 topics, here rectangles.



**Figure 24.** ROS2 nodes and topics communicating in M2O2P

AC communicates with Webapp by publishing application controller output, task information, thresholds, and test information if testing mode is set to ON. Furthermore, AC subscribes to topics when the user communicates through UI with the AC, here in the cases of calibration, testing the different gestures, the status change through UI and filtering and testing mode changes.

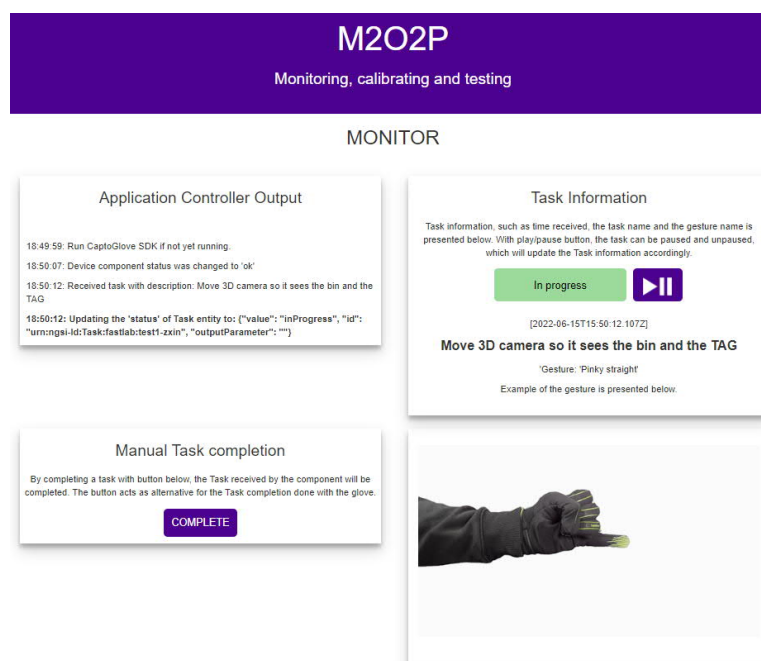
When AC is started, the /component\_status is used to update the status to ok. If there is new task for M2O2P, the task will be published to /taskname topic. The task status will be updated through /task\_completion topic, and the command identifier is updated upon task completion using /command\_id topic. If the task is completed by some other component, /status\_completion\_time is used to notify AC about such action. In all the before said messages the message type is string, and in the cases where the message contains various information, the message is sent as JSON format serialized from Python dictionary.



Additionally, AC sends the sensor values and sensor states to ROS2 topics, that are forwarded to ROS1 topics via Integration Service. This way M2O2P can be utilized as interface for forwarding the sensor information to some other application that needs it.

### 5.1.3 Web User Interface

Web User Interface was developed for M2O2P to be supportive Graphical User Interface (GUI) and offers communication between the component and the user. The interface was divided to four sections. The first section offers monitoring tools, such as output of AC, task information and manual task complete option through the GUI. The monitor section of the UI is illustrated in Figure 25.



**Figure 25.** Monitoring section of the GUI

The output of AC acts as outlet and provides information about what is going on the background when the component is in running state, such as notifications to hold gesture or that the user is making a wrong gesture. The task information will hold the description of the task, and under it the gesture GIF shows what gesture to use for completing the task. In a situation where the glove cannot be used to complete the task, it can be completed with specified *complete* button.

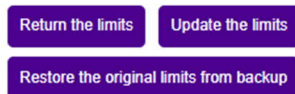
The next section of the UI provides the ability to calibrate the glove. There were two thresholds set for the glove, upper and lower threshold – bent and straightened threshold. These thresholds, per finger, are configurable through the UI by using the buttons shown in the Figure 26. The calibration section can be used on runtime, and the thresholds are changed immediately.

## CALIBRATION

Change thresholds for fingers if the application don't recognize your bent/straight fingers or it recognizes them too easily. Changes takes place immediately at runtime.

With buttons below the table, you are able to return the limits to the state before adjusting them, update the limits, which will update the original\_limits.txt file accordingly or restore the original limits from backup, which will restore the limits that were given to the application when it was started (using the original\_limits.txt)

	Straight threshold		Bent threshold		Current values
Thumb	+100	-100	+100	-100	[1000, 800]
Index	+100	-100	+100	-100	[600, 250]
Middle	+100	-100	+100	-100	[1200, 500]
Ring	+100	-100	+100	-100	[1800, 800]
Pinky	+100	-100	+100	-100	[2600, 1300]



**Figure 26.** Calibration section of the UI

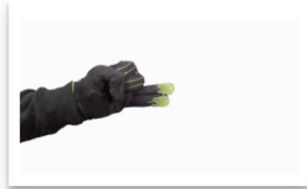
Testing section of the UI consists of table and GIF of the example gesture that user picked. The main function of the section is to provide an ability to user to train the gestures, test out the glove functionality and combined with the calibration section, the glove can be easily calibrated. The testing section includes a table, where there are two rows, one for values that the user is doing, and the desired values that the user tries to replicate. The first column holds the raw sensor values, the second holds the states and the last holds the gesture. The dropdown menu can be used to pick a gesture, which the user then tries to replicate. Testing section used with the calibration section provides powerful tool to calibrate the glove and seeing the effects on the calibration on runtime. After the testing section, additional options section offers the user to the ability to toggle the filtering by the incoming task off or on. These two sections are illustrated in Figure 27.

## TEST

Activate and Deactivate testing mode with the buttons below.

When testing mode is activated, you can change the desired gesture, which will show you the desired states of fingers. After that you can try to replicate the states. It is beneficial to have testing mode on when calibrating the glove. This way you are able to see what fingers causes the problem, and then refine the limits of that finger. Additionally, all the gestures can be seen as GIFs by changing the Desired Gesture from dropdown menu.

	Raw sensor data	States of fingers	Gesture
User	[2735, 2122, 3404, 3223, 3933]	[0, 0, 0, 0, 0]	none
Desired		[2, 2, 2, 0, 0]	Ring and pinky straight



## ADDITIONAL OPTIONS

Filtering Activation

Use Switch filtering mode button to change the filtering mode ON/OFF. Filtering mode affects if the commands sent forward are filtered by the received tasks, or if all the commands are sent forward. In normal behavior, the filtering mode is ON, since it secures reliability of the M2O2P.

**Figure 27.** Testing and additional options section

### 5.1.4 ROS2-FIWARE bridge

The communication between ROS2 and FIWARE has been taken into consideration in other projects, and bridges between these exists, such as Integration Service [110] and FIROS2 [111] that are both created by eProxima<sup>C</sup>. However, both of these options only supports NGSIv2 information model [112]. Since in this Thesis the NGSI-LD information model was used, there was a need for new bridge. The communication between the FIWARE and the module was established with requests -Python library.

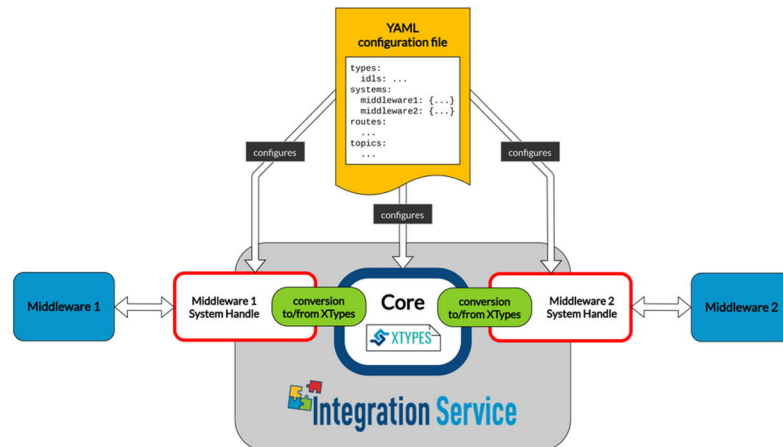
ROS2-FIWARE bridge publishes and subscribes to ROS2 topics to communicate with AC and creates a socket server, which is set as an endpoint for FIWARE subscriptions. At the start up, the bridge will create the necessary subscriptions.

From AC to FIWARE, there are three topics that are handled: `/component_status`, `/task_completion` and `/command_id`. `/component_status` topic is used by AC to provide updates on M2O2P status, which is updated to the Device entity as explained earlier either *ok* or *shutdown*. `/task_completion` topic is used by AC to provide new statuses for the Task entity, which would be either *paused*, *inProgress* or *completed*, and `/command_id` topic is used update the *commandId* attribute of the Device entity.

From FIWARE to AC, there are two topics used, `/taskname` and `/status_completion_time`. The `/taskname` topic will be used to provide needed task information to AC when the task is received through the subscription notification. The `/status_completion_time` topic will be used to provide information for AC if the ongoing task is completed by another component and is made as precaution for such event. Such event won't happen in the solution of this Thesis.

### 5.1.5 Integration Service

When researching about eProxima<sup>C</sup> Integration Service [110], the capabilities of the it was realized as a ROS2-ROS1 bridge. Since the M2O2P offers a direct communication method with ROS2, with the Integration Service the same information was able to be communicated to ROS1 topics. Integration Service is “middleware of middlewares”, which enables communication between two different communication methods. The architecture of Integration Service is presented in Figure 28.



**Figure 28.** Integration Service architecture [110]

Integration Service is used to forward the command identifier at task completion, and the sensor values and their states, to ROS1 topics. Integration Service requires a mapping file, for it to know what topics to listen in ROS2 side and to what middleware forward the information. This information is provided as Docker volume to the Integration Service.

The Integration Service is not used in the final solution, since the more modular solution was achieved using the FIWARE as middleware. However, it acts as an option for the created solution, and is part of the component, and therefore was mentioned in this Thesis.

## 5.2 Orchestrator Application

The FIWARE API of ROS2-FIWARE bridge was reused in the implementation of OA, and the logic behind the task assignment and proceeding in the process was created on top of it. OA has three main functions what comes to process management, assign tasks when the earlier one is completed, handle possible input data for each task, and handle and save possible output data for each task. The process is given to OA in JSON file as Docker volume and holds the necessary information for OA to know how to proceed in the process. Example task included in JSON file is presented in Figure 29. The JSON

entity guides OA to create Task entity, that will tell 3D-module to calibrate using the chessboard center achieved from the robot controller from earlier task and the side that was provided by the human operator in the teaching process.

```
{
  "taskNumber":6,
  "processId":"teachbinpick",
  "involves":"ca-zivid-1",
  "workParameter":{
    "dict":[
      "chessboardcenter",
      "side"
    ]
  },
  "taskName":"Calibrate",
  "taskCode":"Z-CA",
  "commandId":[2],
  "nextTask":[7,1]
}
```

**Figure 29.** Process in JSON format given to Orchestrator Application in teaching process

Here the task holds a task number of 6. At the end of the JSON entity, there is *nextTask* attribute, which is mapped to the task numbers, and depending on if the task was completed with *PROCEED* or *REDO* continuation, and the next task number will be chosen from the list respectively. The *processId* is used to name the process in a case where the process is saved for example in database with other processes. The “involves” attribute holds the device identifier, meaning the identifier used in “involves” attribute in the Task entity. The *workParameter* attribute will be directly put as the value for the *command\_type* attribute of *workParameter* attribute in the Task entity except when the key *dict*. If such case the OA will use the keys provided in the list to find the required attributes from Python dictionary. These attributes are saved from previous tasks. The names of the attributes will be used as keys, and the retrieved values as the values for the attribute, and they will be similarly put as the value of the *command\_type* attribute. The *taskName* holds some human readable description of the task. The value of *taskCode* will be the value of the Task entity *isDefinedBy*. If the entity holds *commandId* attribute which is something other than 0, the number will be added to *command\_type* attribute with a key of *commandId*.

When there is a need to output some information, the name of the key is provided in the *output* attribute. In Figure 30, RCA is guided to output the *chessboardcenter* in the *output*

attribute of *outputParameters* in Task entity. OA will save the returned value to the dictionary with the key *chessboardcenter*.

```
{
  "taskNumber":3,
  "processId":"teachbinpick",
  "involves":"abb-yumi-1",
  "workParameter":{
    "R":"","
    "L":"9",
    "output":"chessboardcenter"
  },
  "taskName":"Output the left chessboard position",
  "taskCode":
  "R-O-LCP",
  "commandId":[0],
  "nextTask":[5]
}
```

**Figure 30.** Process in JSON format given to the taught system to validate 3D-module

## 5.3 Robot Control

The implemented robot code can be divided in two sections, to the RAPID code created for the ABB robot controller (Sec. 5.3.1.), and the containerized Robot Controlling Application (Sec. 5.3.2.) that handles communication with FIWARE and sends the commands through TCP/IP connection for the robot controller.

### 5.3.1 ABB Robot Controller

The ABB robot controller of YuMi was programmed beforehand offline, by implementing flexible yet static functions that can be triggered using the Robot Controlling Application. ABB Robot controllers are programmed mainly in RAPID programming language [113]. The RAPID code is used to create socket server for each hand of the YuMi, and client on RCA connected to the IP address of the robot controller. The robot controller and the laptop where the Docker containers are running are put in same network with Ethernet cable.

There were total of nine functions created in the robot controller:

1. Enable walk-through mode
2. Disable walk-through mode

3. Save current position of the specified robot hand and send it to Robot Controlling Application
4. Close the gripper
5. Open the gripper
6. Go “Home” position
7. Go to RCA defined position using Joint move
8. Go to RCA defined position using Linear move
9. Send chessboard center

The main loop of the program listens to TCP/IP socket connection and responds when and how is needed. Every time there is a new task received, the loop will answer first a string message “procedure started”, then, if necessary, provides the output information in next message, and “procedure finished” once the robot has done the procedure. These functions are called with their corresponding function number. In the 3. procedure the Robot Controller will respond the necessary data of the end effector pose and in 9. procedure provides the chessboard center saved earlier in the robot controller, that is used for the calibration and finding the grasp pose in 3D-module. In 7. and 8. RCA will provide the target for the robot.

There were two supportive functions created to send and receive robot targets: one which changes a robot target information to string format, and one that reads the string and saves the information as robot target. These functions enable the saving of the robot targets to RCA instead of the memory of robot controller.

The Yumi robot is equipped with ABB Smart Grippers [92]. For the smart grippers there are special commands to be used to control them in the RAPID code, such as *g\_gripIn* and *g\_gripOut* for closing and opening the fingers of the gripper. These commands are utilized in the functions that closes and opens the robot grippers.

### 5.3.2 Robot Controlling Application

RCA handles three types of tasks, static tasks, static tasks that requires output from the robot controller and tasks that includes moving from current position to provided position. Static tasks are sent to the controller providing only the procedure number, and in case where output is required, RCA will expect additional message that holds this specific output value. The output can be current pose of the end-effector, which is saved to a dictionary inside RCA with a pose name provided by OA, or other value that is passed to the OA, such as the center of the chessboard tag. In a case where the robot needs to

move to a pose, the OA provides pose name for RCA, which then finds the corresponding pose from the dictionary saved before and provides the pose for the robot controller. Tasks that require robot movement also includes an offset information, which is useful when moving the robot for example over some position.

## 5.4 3D-module

3D-module was created having the same FIWARE interface as the other software component and is deployed as Python script running on the host PC. In the development phase the module was tested in Docker container, and it was discovered that the functions take at least double the time when the module is containerized. Following subchapters will present the functions created for the application: acquisition of the point cloud is presented in Sec. 5.4.1, camera location calibration function in Sec. 5.4.2 and function that finds the grasp pose for the use of the YuMi is presented in Sec. 5.4.3.

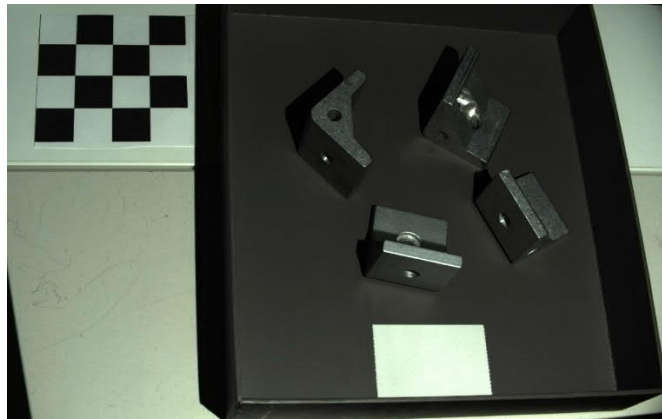
### 5.4.1 Acquire PCD

Other functions of 3D-module requires the acquired PCD from the Zivid camera, hence this function was created. When testing the acquisition of PCD, following acquisition settings were chosen to be used in the implementation since they gave best results:

- Exposure time: 6500ms
- Aperture: 7.15
- Gain: 1.3
- Brightness: 1.8

Furthermore, the outlier, noise, and reflection removal and gaussian smoothing was enabled from the settings. On top of acquiring a PCD, the function will transform the point cloud to 2D image for the calibration function. An example of the 2D image captured and with the camera is presented in Figure 31.

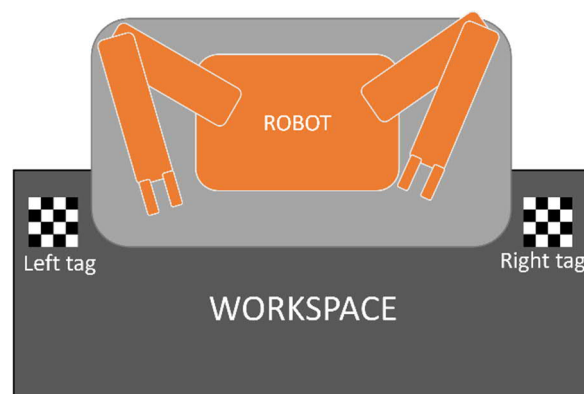




**Figure 31.** Example of 2D image captured by the Zivid 3D camera

### 5.4.2 Calibration

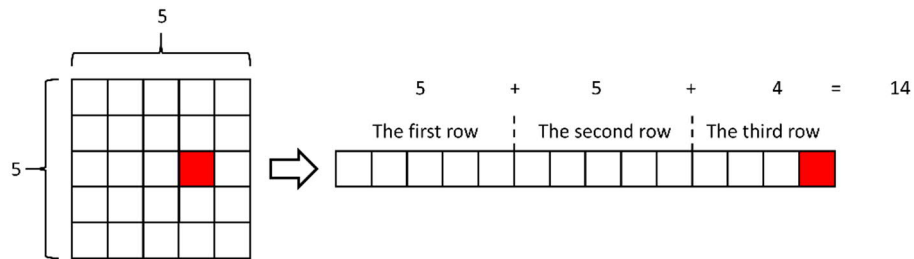
The idea behind the whole process is to redefine the use of the robot cell, meaning for instance that the camera location might change. The calibration in the means of finding the location of the camera and ultimately the points seen by the camera in the robot frame is primary objective to make the system work. To find the camera location in the scene, there needs to be some point in the scene that is visible to the camera and is known point for the robot controller. Furthermore, the used point should also provide information about the orientation, so the PCD can be transformed from camera frame to robot base frame. There was 4x4 chessboard tag used for this purpose. There are chessboard tags in the left and right corner of the workspace, illustrated in Figure 32. The idea would be that these tags are in every robot cell where the Bin-Picking operations are needed.



**Figure 32.** Workspace from top view with chessboards in the corners

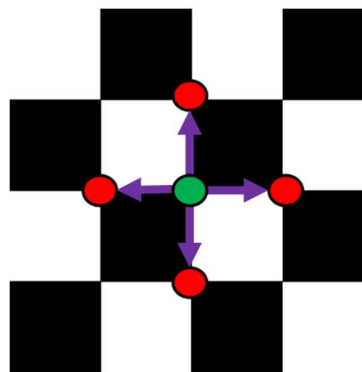
OpenCV Python package [114] provides a function that finds chessboard corners which is here used to find the pixels corresponding the chessboard corners from the 2D image. The function will convert the image to black and white and generate quads to find the squares from the image [115].

The goal of the calibration function is to calculate the transformation matrix  $T_{ca}^r$ , the camera frame with respect to robot base frame. At the start of the function the chessboard corners are extracted. The calibration function knows from human input, which side the camera is from the robot's perspective. Furthermore, the location of the chessboard tag on the table is given as output from earlier task, and as input for the calibration task. The PCD is saved in one dimension, yet in order, and the corners found from the 2D image holds the information in two dimensions. To find the corresponding point in point cloud, the index of the chessboard corner needs to be calculated as follows (Figure 33).



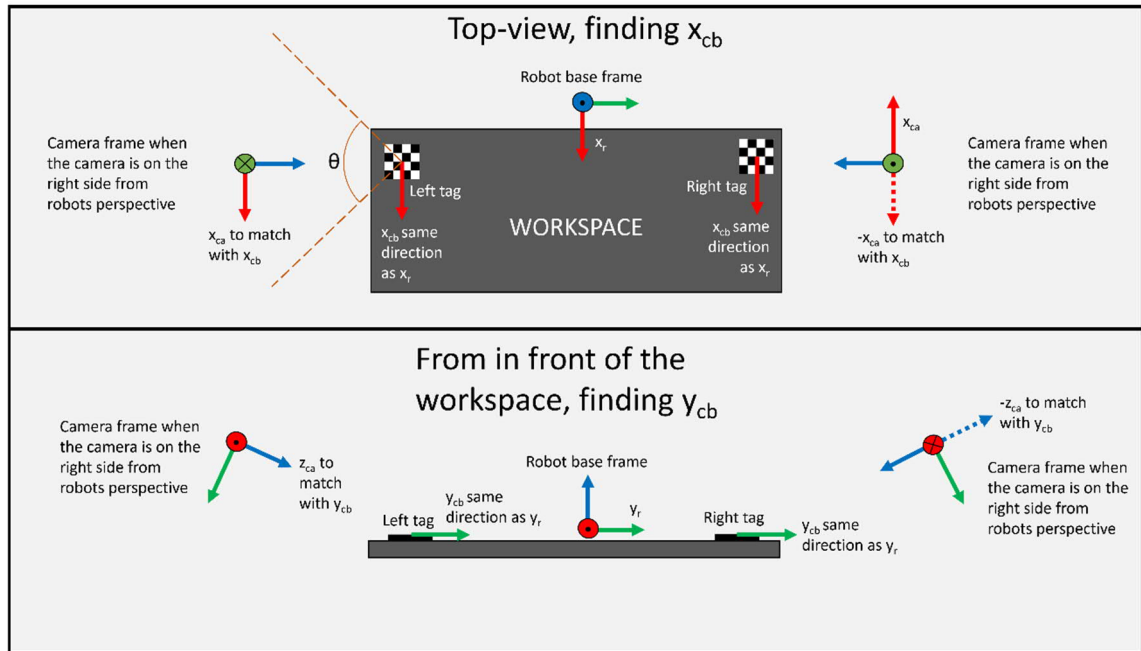
**Figure 33.** Calculating the index of the corner point in point cloud

As the figure illustrates, if in a 5x5 image the corner pixel would be the red one, the corresponding index in the point cloud would be 14. For each corner, the corresponding 3D point is calculated, and then 3D vectors are calculated from the middle point to the corner points, referred here as  $v_1-v_4$ . Only the horizontal and vertical corner points are included, as illustrated in Figure 34.



**Figure 34.** The middle point (on green color), side points (on red color) and the 3D vectors between the middle point and the side points (in purple color)

If the chessboard is looked at from directly top of it and from the robot's side of the table, the vector to the left matches the robot frame y-vector  $y_r$  and the vector that goes up matches the robot x-vector  $x_r$ . This is illustrated in Figure 35.



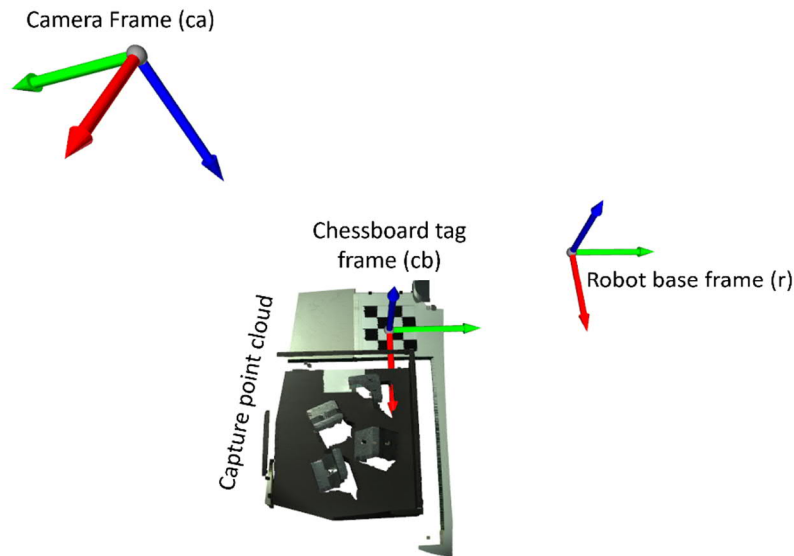
**Figure 35.** Finding the base tag frame using the camera frame

When camera is located on the right side of the robot (on the left side in the image), the vector from  $v_1-v_4$  that has smallest angle to camera frame x-vector  $x_{ca}$  will be interpreted as the x-vector of the chessboard frame  $x_{cb}$ . Similarly, the vector that has smallest angle to the camera frame z-vector  $z_{ca}$  will be interpreted as  $y_{cb}$ . When the camera is on the other side, same vectors from the camera frame are respectively  $-x_{ca}$  and  $-z_{ca}$ . Using these two vectors, the normal vector can be calculated with cross product. Since the camera is pointing down, the normal that has larger angle compared to the  $z_{ca}$  will be chosen as  $z_{cb}$ . After this, it is certain that the  $x_{cb}$  and  $z_{cb}$  has 90-degree angle between them, yet it is possible that the  $x_{cb}$  and  $y_{cb}$  do not have exactly 90-degree angle between them. Therefore, the  $y_{cb}$  is calculated similarly as  $z_{cb}$ , by calculating the normal vector for  $x_{cb}$  and  $z_{cb}$ , and using the normal that has the smaller angle compared to the earlier calculated  $y_{cb}$ . Now there is a “frame” that matches the robot base frame orientation. Then, the SciPy spatial transform “align vectors” method is used to find the rotation matrix between the camera frame and the chessboard frame. The function utilizes Kabsch algorithm named after the creator, that finds the best rotation between two sets of vectors [116]. The function finds rotation between frames  $A$  and  $B$ . To do this, the function will minimize following loss function to solve the rotation matrix  $T$ ,

$$L(C) = \frac{1}{2} \sum_{i=1}^n w_i \|a_i - T b_i\|^2 \quad (1)$$

where  $w$  is the weight that can be given for each vector and  $a$  and  $b$  are the vectors in the two sets. The weights are not used in this implementation, meaning that the weight of each vector is equal. The algorithm outputs the rotation matrix between the camera

and the chessboard frame, e.g.,  $T_{cb}^{ca}$  without translation. Since the transformation of chessboard frame with respect to camera frame is known, the camera frame with respect to chessboard frame is the inverse of the same transformation and calculating this the  $T_{ca}^{cb}$  can be formed. To transform a PCD from camera frame to robot base frame, the  $T_{ca}^{cb}$  needs to be applied to the PCD, and after that the final translation  $T_{cb}^r$ , the chessboard with respect to robot base frame is applied. The frames and the transformed PCD are illustrated in Figure 36. Applying the transformation at this stage helps in validating of the calibration.



**Figure 36.** *The camera frame, chessboard frame, robot base frame and the point cloud transformed to the robot base frame*

The calibration function assumes that the camera is relatively straight on the left or on the right side of the robot pointing to the robot's direction. The maximum angle from the tag ( $\theta$  in Figure 35), where the camera can be located, forms a 90-degree cone on x-y-plane. However, since the angle is not definite, there might be failures in the calibration with too high angle. This is evaluated in the tests created for the application.

### 5.4.3 Find grasp pose

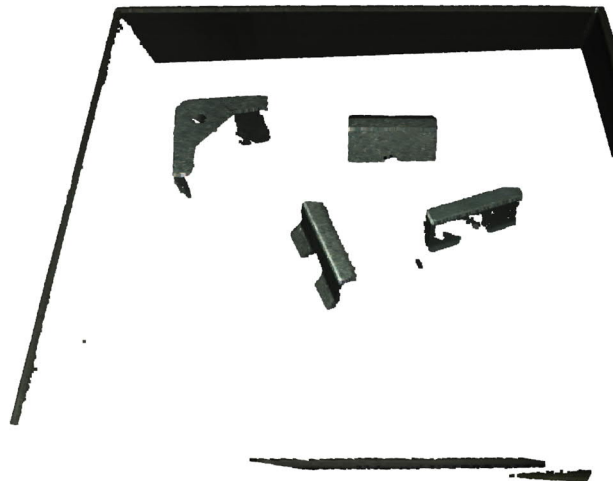
This function processes the PCD and tries to find a grasp pose for the YuMi robot. The function will output the grasp pose in a form of array of arrays,  $[[x,y,z],[q1,q2,q3,q4]]$ , where the first array holds the cartesian coordinates x, y and z, and the latter one the values of the quaternions. Quaternions were used here since they are used to present the orientation information in ABB Robot controllers.

At the start of the function, the PCD is loaded and downsampled using voxel down sample -method. The voxel downsampling means that there is voxel grid created that can be

thought as 3D boxes in 3D space. Inside each voxel the points will be presented through the centroid of the points and will be downsampled to the density where the points will approximately have the voxel size in between them [117]. The function uses 0.5mm as voxel size which will considerably ease the computational times that the function takes yet leaves enough information to calculate the grasp pose accurately.

For later collision avoidance evaluation, the nearest neighbours for each point are found by creating the K-nearest neighbours tree for the point cloud using FLANN library [118] (Fast Library for Approximate Nearest Neighbours) that is included in the Python bindings of the Open3D library.

To further cope with the excessive data in the PCD, it is segmented using Open3D built-in RANSAC function to separate the ground plane (here the table surface) from the PCD. The output after these steps is presented in Figure 37.



**Figure 37.** *Downsampled point cloud without the ground plane*

The point cloud will be then transformed to the robot base frame similarly as at the end of the calibration function.

The PCD is filtered, by calculating the mean of all the points in the PCD and deleting all the points that are further than 150mm away from the mean. After this, the DBSCAN algorithm [85] provided in the Open3D library is used to cluster the PCD. The algorithm is fast, robust, and gave best results when deciding what algorithm to use to cluster the parts. The function will then delete the noise points, labelled as -1 by the algorithm. If the DBSCAN will provide under five clusters, it means that the PCD was not clear. In this case, the new PCD is acquired, and the calibration is done again. This logic was created in the process that OA uses. If there are more than five clusters, there are multiple loops that will process the point cloud in the way, which leaves only the parts to the PCD:

- The clusters that have under 500 points are deleted
- The clusters that have points over 40mm away from the mean of that cluster are deleted

Sometimes the cluster will have small number of points, which indicates that the cluster is not a part. Sometimes the bin might stay amongst the clusters, or additionally some other part, such as table corner. These clusters are deleted with the second filtering loop. After the excessive clusters are excluded from the set of clusters, the clusters are put to an ordered list, which has the largest cluster at first, and the smallest at last. The rest of the procedures are done only to the largest cluster, and if the procedure fails to find the grasp pose using the largest cluster, the loop will continue to the next largest cluster. Example of the found clusters are presented in Figure 38.

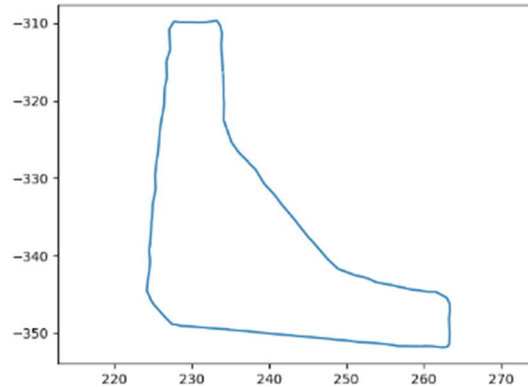


**Figure 38.** Found clusters painted with their corresponding colour

First, the normals are calculated for the points in the cluster. This information is used later to find 90-degree edges from the PCD. The KDtree for the cluster is then found similarly as was found for the whole point cloud. Using the nearest neighbours, the statistical outlier removal algorithm provided in Open3D library is used to remove outliers from the cluster. The chosen values for the function were 20 nearest neighbours, and the standard deviation ratio of 2.0. The nearest neighbour amount specifies how many points are considered when calculating the average distance to a specific point, and standard deviation ratio will set a threshold level for the standard deviation of average distances in the point cloud. More aggressive filter for removing outliers is achieved by lowering the threshold [119]. The algorithm then executes loop to find the topmost point, and with that, all the points that has maximum -5mm height from the topmost points are extracted and further processed.

The topmost points are saved in temporary variable, and the z component is deleted from all the points. Now the points present as 2D plane. From this 2D plane the boundary points are extracted. Boundaries are extracted by finding alpha shapes [120] with the

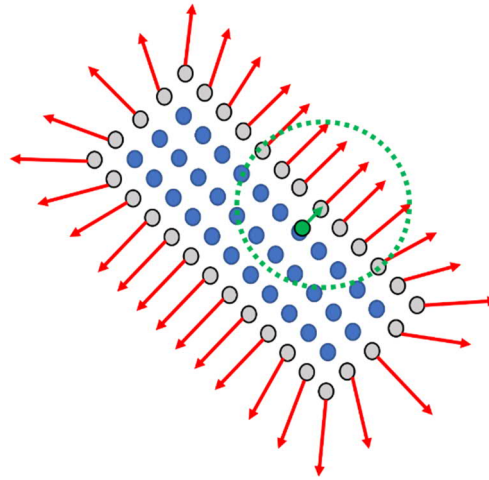
help of Delaunay triangulation [121] that is provided in SciPy spatial library, which uses the Qhull -library to compute the tessellation [122]. Example of the boundary found is presented in Figure 39.



**Figure 39.** Line composed of the boundary points for a part

For each boundary point, the KDtree is used to extract the neighbours in 3mm radius and the angle between each normal and  $z_r$  is computed. The amount of normals, that have angle between 80- and 100-degree is counted. If the counted value for the current boundary point is over five after all points are evaluated, the boundary point is considered as corner point and the index of the current boundary point is saved.

Next for each boundary points, the mean of the points in 6mm radius is calculated. Vector is formed from the mean to the boundary point, which is then set as the normal of that boundary point, but with z component of the vector as zero. After this, from top view, the normal could look like as illustrated in Figure 40.

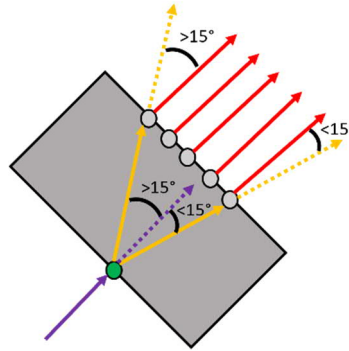


**Figure 40.** *The gray points are the boundary points, blue points are non-boundary points, the green dashed line acts as neighbour radius, green point as the calculated mean, green arrow as the vector that drawn from the mean to the boundary point and red arrows illustrates the normal aligned with the vector*

The purpose of this procedure is to provide normal that points away from the other points of the PCD, and to use them when finding if aligned pairs, which is explained later. Furthermore, this procedure will make the hidden side of the part “known” by assuming it will hold a 90-degree angle.

Since the function assumes that there are 90-degree angles, in case where the corners are not found would imply that one face of the part on top of top side is facing directly to the camera. In such case the mean of the topmost points will be the pick position, and the x-vector of the grasp pose  $x_{gp}$  is aligned to same direction as the  $x_{ca}$ . In the case where the corners are found, for each corner point, loop the boundary points, and calculate angles between negative of the normal of the current corner point, the vector that is created from the current corner point and the boundary point in question, and the normal of the boundary point in question. This procedure is illustrated in Figure 41. The vectors can have maximum 15 degrees angle between them, and the points can be maximum of 42mm away from each other (the maximum distance between the gripper of the ABB robot).





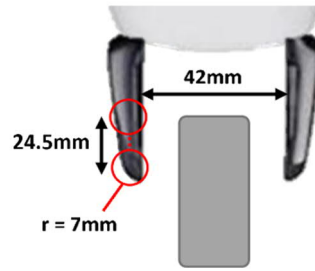
**Figure 41.** *The gray box is the part, the green point is the boundary point, the purple arrow is the negative of the normal for that point, purple dashed arrow is the same vector but taken next to the yellow candidates for the evaluation of the angle, similarly the yellow dashed lines are taken for the evaluation. The lower candidate would present as aligned pair, but the top candidate doesn't, due the angles that they have*

When the algorithm finds a pair, it calculates a cost for this pair. The cost of the pair consists of two components, the distance between the points, and mean of the distances between both points and the center of the topmost points. These components are summed together to form the cost for the current pair under inspection. If the topmost points consist of points A, B and C, the points A and B would be the pair in question, and the *dist* would be the distance between the points, the simplified cost function would be as follows:

$$cost = dist(A, B) + \text{mean}(dist(\text{mean}(A, B, C), A), dist(\text{mean}(A, B, C), B)) \quad (2)$$

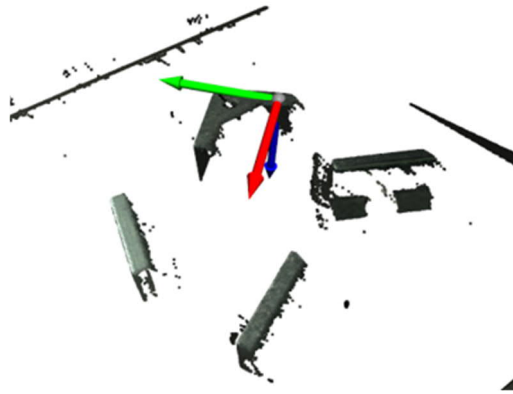
If the cost is lower than the cost calculated on earlier iteration or it is the first cost to be calculated, the pair will go under inspection of collision avoidance. The cost will favour pairs that have points close to each other and close to the center of the topmost points, which will intuitively lead to better grasp pose.

Using the earlier calculated nearest neighbours' tree for the whole PCD, the function will make sure that the gripper fingers will not collide with the environment during the grasping. The function will calculate two vectors that are 28mm long and are pointing from the center of the chosen two points to each of the points. This will be the center of the gripper finger since the distance from the gripper frame to the inner side of the gripper is 21mm and the gripper finger radius is approximately 7mm. From these points, and points under them, the function will check if there are points inside 7mm radius. The check will be done 7 times, each time 3.5mm lower than in last iteration and the furthest sphere will be located 24.5mm below the top sphere. This collision avoidance procedure is further explained using the image of the gripper and the red spheres in Figure 42.



**Figure 42.** Collision avoidance method

If the collision avoidance check is successful, and the best pair is found, the x-vector of grasp pose  $x_{gp}$  is determined by calculating the angle between the vector from the mean to each point in selected pair, and the  $x_r$ . The vector with smallest angle will be chosen as the  $x_{gp}$ . This aims to find repeatable results by always setting  $x_{gp}$  to relatively same direction. Example of grasp position found using the algorithm is provided in Figure 43.



**Figure 43.** Example grasp pose found for the robot

If pairs are not found, the algorithm will continue to the next cluster. If all clusters have been looped and yet no grasp pose has been found, the function will provide info for the orchestrator that either the bin is empty, or there is a need for new PCD. In the teaching procedure, the orchestrator will take the new PCD, since there must be parts still, and not finding any part means that the PCD was not good enough for finding the grasp pose.

## 5.5 Testing Procedures for the System

The implemented solution will be evaluated using testing procedures. Testing procedures will partly answer the research questions, and the results of the tests will be presented in Results -chapter. The tests created for this Thesis focuses on functionalities of the system and will be conducted by the author of this Thesis. Tests can be categorized in three different categories, isolated tests for M2O2P (Sec 5.5.1), isolated tests for 3D-module (Sec 5.5.2) and test scenarios created for the complete Use-Case (Sec 5.5.3). OA, RCA, and robot controller are tested in Use-Case tests.

The Bin-Picking operations will be evaluated using only one type of parts, which are used in assembly of the robot cells in FAST-laboratory. Image of one of these parts is presented in Figure 44. The testing of the system will focus on scenarios where the parts are semi-structured – in random poses, but not on top of each other.



**Figure 44.** *Parts used in evaluation of the Bin-Picking operations*

### **5.5.1 Isolated tests for M2O2P**

The reliability of M2O2P is tested two ways, by finding out if accidental gestures are made during object manipulation and figuring out how well the application can recognize the gestures. Accidental gestures are evaluated using grasp taxonomy of Cutkosky [123] and evaluating the accidental gestures when handling the objects in the Use-Case. The grasp taxonomy provides comprehensive testing setup for accidental gestures caused by different styles of grasping. To simplify and visualize the grasp taxonomy, the taxonomy was recategorized under their corresponding object, and is presented in Figure 45.

Object	Style of grip	Demonstrations of the grips
Plate	Push	
Gymstick	Heavy wrap with small diameter and abducted thumb	
Cup	Heavy wrap with large diameter	
Tennis ball	Wrap, hold with fingers and fingers as tripod	
Pen	Wrap, holds with thumb + 1-4 fingers	
Coin	Lateral pinch	
Jar cap	Wrap and hold with fingers	

**Figure 45.** Cutkosky's grasp taxonomy adopted with testing objects

Each of the grasps are held for 20 seconds, and in the meantime the grasp itself is re-adjusted to find if the application registers the current grasp as gesture. On top of the specific shapes, the accidental gestures are tested in the teaching procedure in similar way. The object manipulation tasks that the operator needs to complete are handling the robot arm and moving the camera stand. The robot arm is manipulated with different grasping points, and camera stand is moved back and forth to see if the system registers any gestures.

To counterbalance the test of unwanted accidental gestures, the second isolated M2O2P test is used to find out the accuracy of the gesture recognition, i.e., the wanted behaviour. M2O2P is first calibrated, and all 21 gestures are replicated one by one using the Testing section provided in User Interface of M2O2P. This is repeated 10 times, and in between iterations the glove is taken off and put back on. The purpose of this test is to report the accuracy of the application and evaluate the need of calibration with one user when the glove is taken off in between.

### 5.5.2 Isolated tests for 3D-module

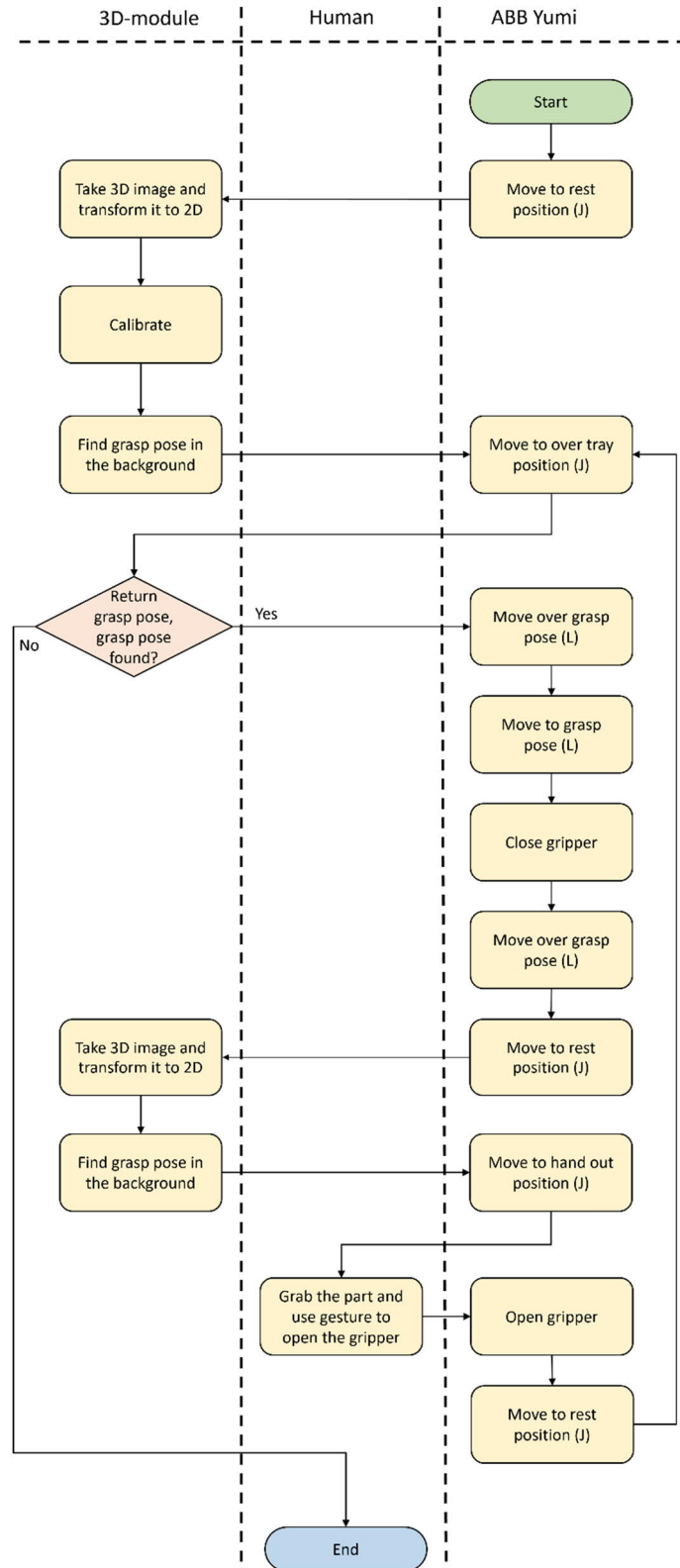
The calibration function created for the 3D-module, which was presented earlier in Sec. 5.4.2., is evaluated by the means of maximum angle where the camera can be located with respect to the calibration tag. The camera will be moved in greater angle, and the result of the calibration will be evaluated using the Point Cloud including the calibrated frame.

3D-module will be further tested by clocking the times each of the function takes and analysing them. Lastly, the collision avoidance function created for 3D-module is tested by providing scenario where the grasp point should not be found, and then where it can be found.

### **5.5.3 Tests done for complete Use-Case**

The teaching procedure is repeated six times, three times using the glove as interaction method for proceeding in the process, and three times using a mouse and the GUI of the application. These tests will be captured on a video for later analysis. Additionally, the time of each task is clocked. The purpose of this testing procedure is to find out how glove-based input compares to more traditional input method using GUI, and to evaluate the overall flow of the teaching procedure.

The testing process created will evaluate the teaching procedure. The robot will pick the parts from the bin and hand them to the human. Tasks of the procedure are clocked and captured on the video for further analysis. Flowchart of the evaluation process is presented in Figure 46.



**Figure 46.** Process created for evaluation

The robot is moved with two types of commands. The first type, which can be seen in the flowchart as L, moves the robot in straight line trajectory with the shortest path be-

tween the start and the end point. The second type is to move the robot with joint command J, which will move the robot the shortest path from start to end from point of view of the joint angles of the robot. Linear movement will have velocity of 50mm/s, and joint based movement 100mm/s. The process will go on until the grasp pose is not found anymore, which means that the bin is empty. The evaluation process will be executed 10 times to evaluate the grasp poses, and overall performance of the taught system.








## 6. RESULTS

This chapter will present the results of the tests and other notes reported during the evaluation. The chapter is divided to three subchapters, tests done for M2O2P in Sec. 6.1, tests done for 3D-module in Sec. 6.2 and tests done for whole Use-Case in Sec. 6.3 Furthermore, the results of Use-Case tests are divided to teaching process in Sec. 6.3.1 and to evaluation process in Sec. 6.3.2.

### 6.1 Isolated tests for M2O2P

M2O2P was tested in two ways, first by investigating if manipulation of objects causes accidental gestures, and second by investigating the accuracy of the gesture recognition. The results of these tests are provided in following paragraphs.

When testing the different styles of human grasps, four accidental gestures were recognized. The results are provided in Figure 47.

Object	Style of grip	Demonstrations of the grips	Accidental gesture
Plate	Push		
Gymstick	Heavy wrap with small diameter and abducted thumb		Thumb and pinky straight
Cup	Heavy wrap with large diameter		
Tennis ball	Wrap, hold with fingers and fingers as tripod		Thumb, index and middle straight
Pen	Wrap, holds with thumb + 1-4 fingers		*Thumbs up, **middle straight
Coin	Lateral pinch		
Jar cap	Wrap and hold with fingers		

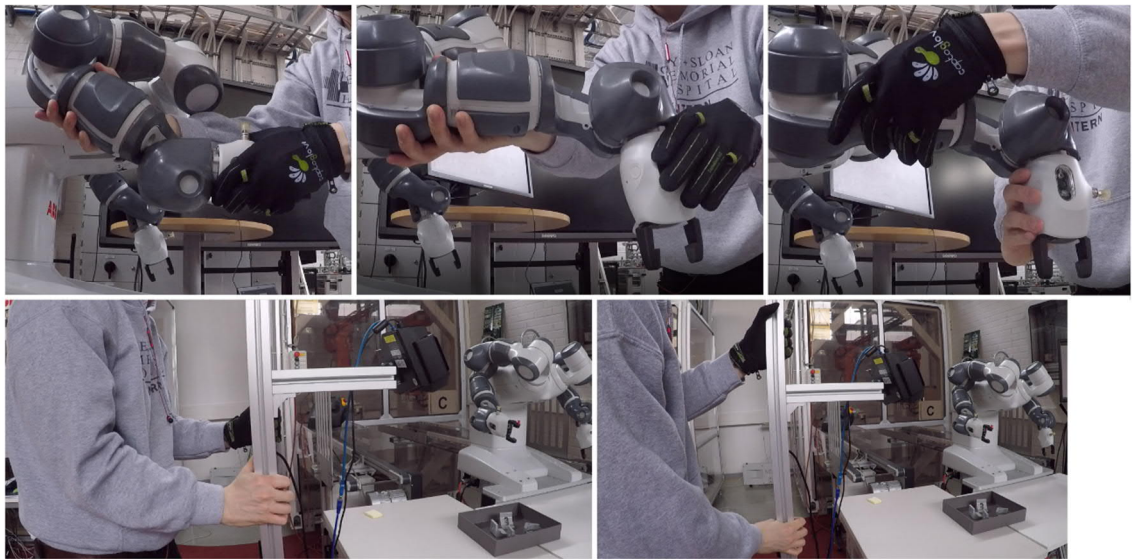
**Figure 47.** Results of tests done using the Cutkosky's grasp taxonomy when wearing the smart glove

Gymstick caused accidental gesture of *Thumb and pinky straight*. The grasp itself seems similar as *Thumbs up*, yet the pinky finger was also considered as straight. When readjusting the grip, the pinky finger was relatively straight at one point, which caused it to be recognized as straight. When holding the ball with fingers as tripod, the *Thumb, index, and middle straight* -gesture was recognized. This can be seen from the image since



those fingers are straight compared to others. When holding the pen, holding with thumb and one finger caused *Thumbs up* gesture, and holding with thumb and two fingers (index and middle) caused the *Middle straight* gesture. Both are surprising when watching the examples of the grasps, yet both were recognized when readjusting the grasps in a way that the bending state of the fingers was close enough it to be recognized as before mentioned gestures. To cope with the accidental gestures, the gestures that are used in tasks that requires such grasps should be chosen to be “harder” gestures.

The grasps needed in the tasks to complete the teaching process were tested to apply the grasp taxonomy for the Use-Case of the Thesis. There are two tasks where objects need to be manipulated when having an ongoing task: moving the stand for the 3D camera and teaching the robot hand new points. The camera stand was moved back and forth with the glove on, and both robot arms were manipulated with walk-through mode on. Both actions were done for each 30 seconds to spot the accidental gestures. The Figure 48 includes images of different grasps and manipulation of the stand and the robot arm.



**Figure 48.** *Examples of the manipulation procedures*

During the tests, the *Pinky straight* gesture was recognized two times in a way that it provided a prompt to the user to hold the gesture, yet it did not send the command forward. The gesture was held for 0.5s for it to be prompted, but not 1.5s which would trigger the sending of the command; hence no accidental gestures were faced in the test.

As the unwanted scenario of accidental gestures was just presented, the second test done for M2O2P evaluated the wanted behaviour, i.e., the accuracy of the gesture recognition. All the 21 gestures that M2O2P offers were repeated 10 times, taking the glove

off in between the iterations, leading total of 210 interactions, and following table was filled (Table 6).

Table 6. *Accuracy of the M2O2P component*

<b>Problematic gestures</b>	<b>Amount</b>	<b>Iteration number(s)</b>
Middle and ring straight	1	4
Index and ring straight	1	7
Amount of succesful interactions: 208		Amount of problematic interactions: 2
Accuracy of gesture recognition: 99.05%		

In this test scenario, the gesture was done and once it was thought to be correct, the UI was checked if the gesture was recognized. All the gestures were successfully recognized, yet in two instances there were problems where one of the fingers needed slight readjustments. If these problems are thought as failed recognitions, the accuracy of the recognized gestures was 99.05 %. However, in a real situation where the UI is visible for the human operator, the operator can see from the UI that the gesture was not recognized and can readjust the fingers, so it is recognized correctly. In both gestures where problems were faced, the ring finger was one of the straight fingers, and it was also the one that needed readjustments. Such problematic interactions can be faced due to poor calibration, which can be corrected with the runtime calibration section of the UI.

## 6.2 Isolated tests for 3D-module

The first test done for 3D-module was to evaluate the largest angle the camera can be in x-y-plane with respect to chessboard tag. For this test there was a specific code added for in the calibration function that will calculate the angle between the  $x_{ca}$  and  $x_{cb}$  and PCD was visualized at the end of the calibration function to find if the calibration was indeed successful. The maximum angle measured between the vectors was 45.34 degrees, which would be 90.68 degrees if applied two times. The higher than 90-degree angle was measured since the camera was not fully straightened. The maximum angle was therefore as expected earlier in the Thesis in Sec. 5.4.2. To be sure that the calibration works, the buffer should be left on the angle, so the lower than 45-degree angle can be seen with naked eye.

The times of each 3D-module function were clocked five times, when moving the camera and scrambling the bin of parts in between. The clocked times and mean time for each function are presented in the following table (Table 7).

Table 7. *Completion times and their mean for each 3D-module function*

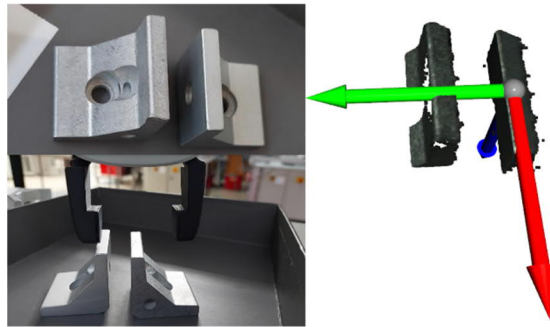
<b>Iteration</b>	<b>Acquire PCD (s)</b>	<b>Calibration (s)</b>	<b>Find grasp pose</b>
1	4,93	0,69	5,14
2	4,51	0,71	4,97
3	5,41	0,69	4,91
4	5,07	0,68	4,24
5	5,49	0,80	4,46
mean	5,08	0,71	4,74

The time that it takes to acquire a PCD is quite long, which can be problematic in process where the system needs to act fast. Most of the time is consumed in saving the Zivid point cloud object to a file point cloud file (.ply) and converting it to 2D. This would need to be considered in runtime by capturing the new point cloud when the robot is not in between of the camera and bin and doing rest of the procedures in the background. However, this function was not added to 3D-module and can be considered as future development topic.

Calibration function was quite fast, and since it must be done only once at the start of the process if the camera has moved, the time is acceptable. Due the fast calibration, it makes sense to do it at the start of every process to be sure that the camera has not moved in between.

Finding the grasp pose took similar amount of time as acquiring the PCD, and since the whole function can be done in the background, there was two new functions added to 3D-module, one which finds the grasp pose in the background, and one which returns the found grasp pose. This will drastically lower the time that the cycle takes. These functions are tested when evaluating the taught system in the evaluation process.

The collision avoidance system was tested, and it was found that it works as intended. Following Figure 49 has an image on the left, where the grasp pose is not found, and on the right point cloud and grasp pose that was found. On the left side, in addition to the top side of the part, only one side is facing the 3D camera, and the parts prevents finding collision-free grasp poses for each other. On the right side, the parts are in a way, where two sides of the part are visible for the camera, and therefore the system can find the grasp pose.



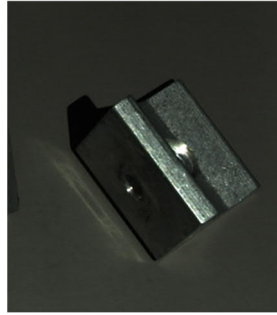
**Figure 49.** *No collision free points found with the example of robot finger colliding on the left side, and found grasp pose on the right side*

Few problems were faced during the tests for 3D-module. If the bin is between the structured light source of the 3D camera and the chessboard tag, even if the camera would see the tag, the bin can cast a shadow over the chessboard tag and the calibration won't work. This situation is illustrated in Figure 50. The camera needs to see all the 4x4 tiles, and in this situation the shadow will blend with the squares closest to the bin shadow, and therefore the function that finds the chessboard corners does not work. However, if the image ends by cutting a part away from side of the chessboard tag, the calibration still works. This can be thought as limitation for the bin and chessboard tag placement with respect to the camera.



**Figure 50.** *Bin casting a shadow on the chessboard tag*

When finding grasp pose, there were some problems faced with the distortion of point clouds that are caused by reflection. Such problematic pose of the part is illustrated in Figure 51. The part is aligned with camera and creates reflection of light to the bottom of the bin. On top of this, the whole dark side of the part is distorted, and the points are scattered along the path from the part to camera lens, and therefore the 90-degree angles are not found for this part. To fix this problem, there would need to be further refinement of the camera settings, or alternatively pick parts that are not metallic, which is the major reason that causes such problem due the reflectiveness.



*Figure 51. Part that caused noisy point cloud due the reflection of metallic side*

### **6.3 Tests done for complete Use-Case**

All Use-Case tests were conducted in FAST-laboratory, and with only one participant, the Thesis author. The tests done for the whole Use-Case were done in two ways. The teaching procedure was evaluated, and the glove-based input was compared to GUI input interacted with a mouse in the first test set in Sec. 6.3.1. The second test set focused on assessing the taught system with the evaluation process in Sec. 6.3.2.

#### **6.3.1 Tests done in the teaching process**

The evaluation of task completion times focuses on human and robot tasks since 3D-module was already tested in isolation and will be tested in later evaluation process. The time of each individual tasks were clocked once, and for new iterations only the human tasks were clocked. The human tasks are the only ones that have notable time variation between the test iterations.

As explained in the Sec. 5.5.3. the teaching procedure was iterated six times, three times using the glove as input method and three times using a mouse as input method. Table 8 was created as result of the tests, and the captured video was further analysed. To better compare each of the input methods, the human tasks were timed and separated in two columns. The first column presents the time that the whole task took, minus giving of the input. The second column presents time, where the human operator stops doing the task, as the task itself is completed, and starts to use or reach the input device. This separation was done using the video footage captured from the test case. The third column holds percentage that the interaction took from the whole task. The tasks that are assigned to be done by the human, the percentage that the task itself consumes should be much higher than what the interaction method consumes. In short tasks, for example when answering a question with the input device, the interaction will consume high percent of the time used for the whole task. For each human tasks, the percentage of using the input device from the whole human task was calculated.

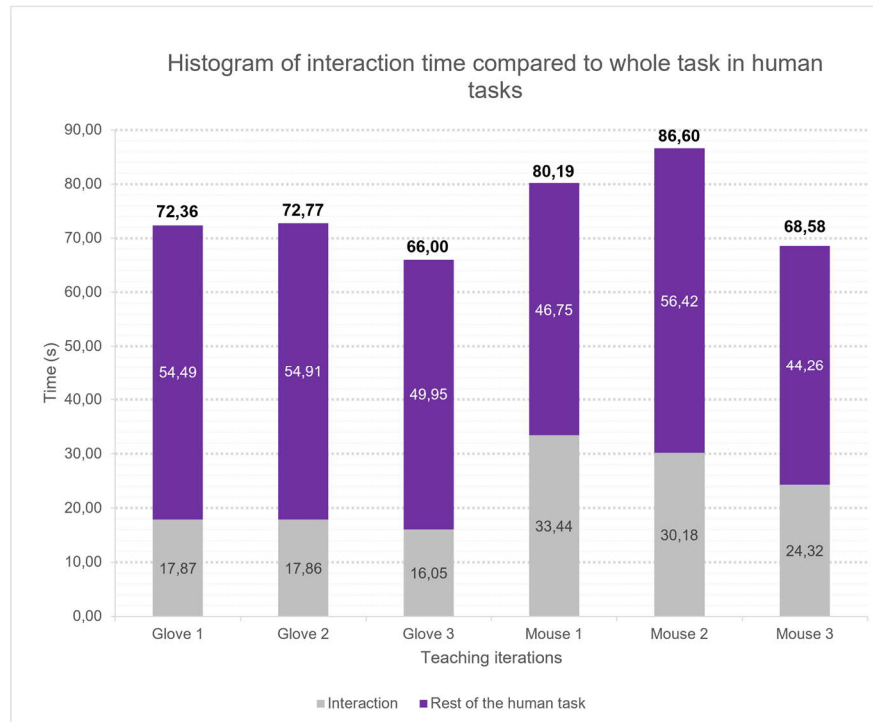
Table 8. Resulting times when testing glove and GUI as input method

Task name	Glove as input method									Mouse as input method								
	Time (s)			Time (s)			Time (s)			Time (s)			Time (s)			Time (s)		
	task	input	%	task	input	%	task	input	%	task	input	%	task	input	%	task	input	%
Move 3D camera	15.40	2.58	14.3	20.22	2.42	10.7	13.30	2.09	13.6	11.40	7.03	38.1	18.66	4.62	19.8	13.30	3.45	20.6
Which side camera is?	3.31	2.32	41.2	2.85	1.98	41.0	2.94	2.32	44.1	4.01	3.98	49.8	4.01	3.98	49.8	3.22	2.97	48.0
Output the chessboard pos.	0.38			0.38			0.38			0.38			0.38			0.38		
Capture PCD + 2D image	5.32			5.32			5.32			5.32			5.32			5.32		
Calibrate	0.69			0.69			0.69			0.69			0.69			0.69		
Find grasp pose	4.42			4.42			4.42			4.42			4.42			4.42		
Bin visible?	6.01	2.15	26.3	5.52	2.13	27.8	7.21	2.24	23.7	5.11	3.21	38.6	7.25	3.99	35.5	5.95	4.15	41.1
Enable walk-through mode	0.52			0.51			0.55			0.52			0.56			0.53		
Teach over tray position	11.33	2.00	15.0	9.42	2.48	20.8	8.98	2.33	20.6	10.23	4.92	32.5	9.77	4.21	30.1	7.02	2.49	26.2
Save over tray position	0.45			0.44			0.41			0.43			0.43			0.46		
Disable walk-through	0.52			0.51			0.49			0.48			0.51			0.47		
Move over grasp pose	2.53			2.53			2.53			2.53			2.53			2.53		
Move to grasp pose	1.06			1.06			1.06			1.06			1.06			1.06		
Close gripper	1.34			1.34			1.34			1.34			1.34			1.34		
Move over grasp pose	1.24			1.24			1.24			1.24			1.24			1.24		
Enable walk-through mode	0.52			0.51			0.55			0.52			0.56			0.53		
Teach rest position	11.05	2.11	16.0	8.82	2.02	18.6	9.26	2.47	21.1	8.76	5.24	37.4	11.08	4.44	28.6	6.10	4.42	42.0
Save rest position	0.43			0.47			0.47			0.49			0.46			0.47		
Teach handout position	5.09	2.30	31.1	6.17	3.92	38.9	4.56	2.19	32.4	4.72	4.32	47.8	3.82	5.13	57.3	5.38	2.89	34.9
Save handout position	0.41			0.45			0.44			0.42			0.41			0.43		
Grab the part	2.30	4.41	65.7	1.91	2.91	60.4	3.70	2.41	39.4	2.52	4.74	65.3	1.83	3.81	67.6	3.29	3.95	54.6
Open gripper	1.23			1.17			1.22			1.41			1.08			1.24		
Disable walk-through	0.52			0.51			0.49			0.48			0.51			0.47		
Go home	4.43			4.43			4.43			4.43			4.43			4.43		
sum	80.50	17.87		80.89	17.86		75.98	16.05		72.91	33.44		82.35	30.18		70.27	24.32	
sum of sum		98.37			98.75			92.03			106.35			121.00			102.98	
Average of sums					96.38									110.11				
Average		2.55	30.0		2.55	31.2		2.29	27.8		4.78	44.2		4.31	41.3		3.47	38.2

The total times of whole teaching sequence, and the average of this total time is presented in red colour. Using the mouse as input method took considerably more time than when using the glove as input method. The total times consumed to teaching are low in general, averaging to 1min 36s with the glove and 1min 50s with the mouse.

Furthermore, the table holds average time for interaction and average percentage that the interaction consumed from the whole human task in green colour. The average time of the interaction was 2.47s with the smart glove and 4.19s with the mouse. On average, the mouse interaction took 69.6% more time than with the glove. For M2O2P to register a gesture and complete a task, the algorithm takes 1.5s. When utilizing natural input methods, here using a glove worn by the user, the interactions can be done relatively fast. The average time from letting go from the robot and registering the gesture for the first time was only 0.97s. Average percentage that using the glove as input device took from the whole human task was 29.7% where the same percentage was 41.2% when using the mouse. This was seen from the video clearly, since the human operator needed to reach the mouse when holding the robot and took more time than just doing the gesture with the smart glove.

To further investigate the differences between the interaction methods, only the times that the human tasks consumed in the whole process were gathered, and following histogram was created (Figure 52). The gray-coloured bars correspond to interaction times and purple-coloured bars to rest of the task. The combined time of human tasks is presented over the bar.

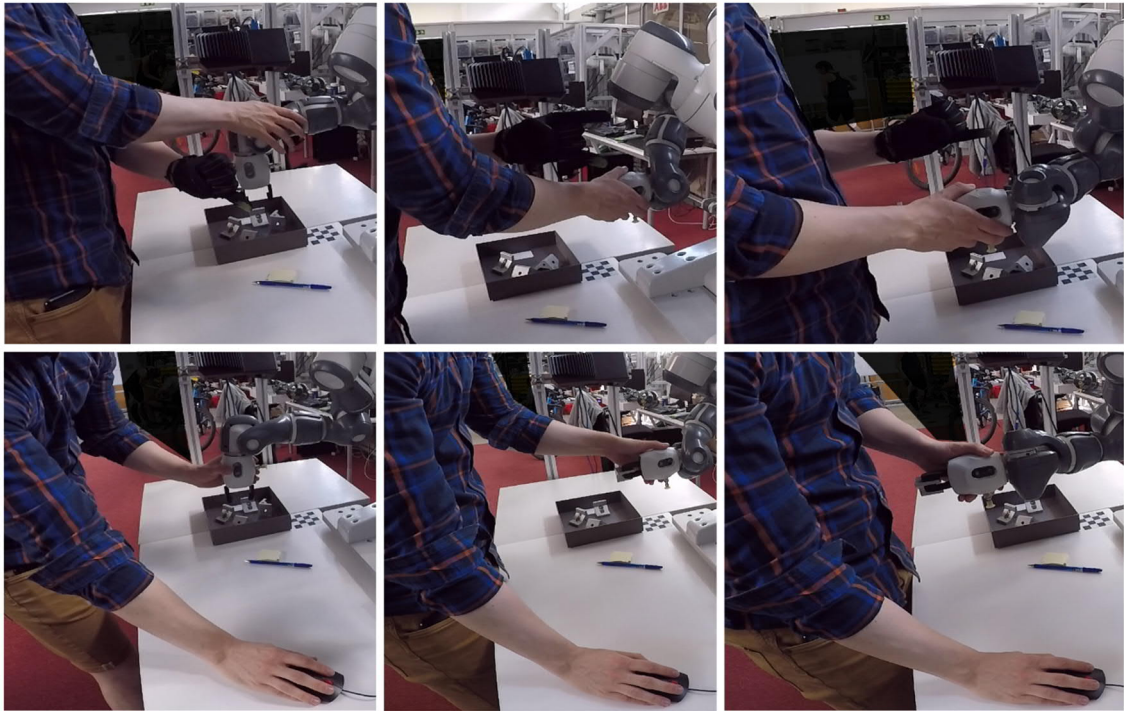


**Figure 52.** Histogram of interaction times compared to whole human tasks in the process

Multiple consecutive human tasks, which don't require human to take the hand off the mouse, would be much faster to complete using the mouse and GUI. However, the tasks usually require physical manipulation by the human, and that's why the glove presents as better interaction device.

The testing scenario is illustrated in Figure 53, where the top row holds the images of teaching all three poses using glove as input, and the bottom row holds teaching the same positions with mouse as input. The glove can be used without moving in the workspace, yet the mouse needs to be reached at the same time taking the focus off from the teaching process, and in worst case the pose that is taught for the robot will change when reaching the mouse. This might be problematic in a situation where the mouse location is further away.





**Figure 53.** *Illustration of teaching procedure using the glove and the mouse and GUI as input method*

The GUI of M2O2P component can be reached in local network using the IP address of the host PC, which in some cases can be located somewhere else, for example in the electrical cabinet of the robot cell. The freedom that is provided using the glove as the interaction method is crucial in such tasks, where the interaction event needs to be done on the spot and moving to interact with the GUI would have negative impact on the flow of the process.

### 6.3.2 Tests done in the evaluation process

To evaluate the teaching process, the evaluation process presented earlier in Sec 5.3.3 was executed, captured on video and times of each task were clocked. The clocked times for four parts (emptying a bin that has four parts) are presented in Table 9. Since the robot movements were kept relatively slow in the evaluation process to spot possible problems, there were two columns added that has 100% faster robot movement. The robot speed was set to 100mm/s in joint movements and 50mm/s in Linear movements in testing case, which would be 200mm/s and 100mm/s after doubling the velocity. The changed values are presented in red colour. The background function for the acquisition of the PCD was not implemented, so the last set of values has the times where the acquisition of the PCD was cut down to 110ms, which is the time that the acquiring of the PCD and processing it takes provided by Zivid [93]. The changed values are presented in green colour.

Table 9. Times consumed in tasks, with added scenarios

Task name	Measurements from one run with three parts (s)				100% faster robot movements (s)				100% faster robot movements and with acquire time that is given in the Zivid manual (s)			
	part 1	part 2	part 3	part 4	part 1	part 2	part 3	part 4	part 1	part 2	part 3	part 4
Move to rest position	4.52				2.26				2.26			
Acquire PCD	5.12				5.12				0.11			
Move over the tray	5.35	5.22	5.25	5.21	2.68	2.61	2.63	2.61	2.68	2.61	2.63	2.61
Move over grasp pose	3.40	3.45	3.92	5.22	1.70	1.73	1.96	2.61	1.70	1.73	1.96	2.61
Move to grasp pose	1.12	1.08	1.11	1.16	0.56	0.54	0.56	0.58	0.56	0.54	0.56	0.58
Close gripper	2.07	1.33	1.58	1.59	2.07	1.33	1.58	1.59	2.07	1.33	1.58	1.59
Move over grasp pose	1.12	1.04	1.10	1.12	0.56	0.52	0.55	0.56	0.56	0.52	0.55	0.56
Move to rest position	6.11	4.57	5.92	6.05	3.06	2.29	2.96	3.03	3.06	2.29	2.96	3.03
Acquire PCD	5.45	5.28	5.31	5.35	5.45	5.28	5.31	5.35	0.11	0.11	0.11	0.11
Move hand out pose	3.36	3.07	2.91	2.41	1.68	1.54	1.46	1.21	1.68	1.54	1.46	1.21
Grab the part	3.28	3.53	3.69	3.23	3.28	3.53	3.69	3.23	3.28	3.53	3.69	3.23
Open the gripper	1.24	1.16	1.25	1.33	1.24	1.16	1.25	1.33	1.24	1.16	1.25	1.33
Move to rest position	2.38	2.24	2.33	2.41	1.19	1.12	1.17	1.21	1.19	1.12	1.17	1.21
Sum	44.52	31.97	34.37	35.08	29.65	20.52	21.94	22.09	19.30	15.35	16.74	16.85

The cycle time from the human using the gesture to release the part, and for the robot to pick new part and bring it to handout position would take approximately 30s. This would mean that the one part that is brought to human should be enough for the human to be busy for 30s. If the robot is used as third hand, such cycle time would be too long for bringing one part to use of the human. In the last scenario, where the robot moves 100% faster and the times that the Zivid takes to capture PCD is cut down to 110ms, the corresponding value is cut down to approximately 13s, which as cycle time is notably better. The system uses the approximately 5.35s, minus the 110ms resulting to 5.24s, to process and save the PCD, and additional 4.74s to find the grasp pose. If these are added together, the resulting time would be 9.98s. The time between the PCD has just been acquired, and the system would need the information about the grasp pose is 10.07s. In such case, the grasp pose would be found just in time. However, this creates limitation for the cycle time and in some scenarios can be too long. To cope with the problem, there would need to be an alternative method for changing the PCD from Zivid format to Open3D format, that does not take so much time.

To evaluate the grasp poses, 10 bins which had four parts in each were emptied. Furthermore, at the start of each iteration the camera location was recalibrated. The successful and failed grasp poses for each part and iteration are presented in Table 10.

Table 10. Accuracy of successful grasping

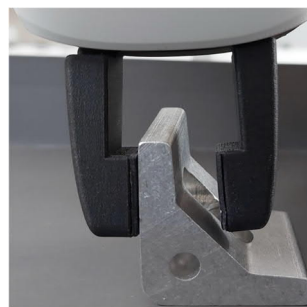
Iteration	part 1	part 2	part 3	part 4
1	o	o	o	o
2	o	x	o	o
3	o	o	o	o
4	o	o	o	x
5	o	o	o	o
6	x	o	o	o
7	o	o	o	x
8	o	o	o	o
9	o	o	o	x
10	o	o	o	o

Successful grasps: 35, failed grasps: 5

Accuracy of grasping: 87,5 %

o = successful, x = failed

The accuracy of the grasping was 87.5 %, which is not too poor, yet should be higher, especially when the parts are brought to human. However, the grasping accuracy was affected by the limitations of the robot in three grasp poses. On iteration two, the joint angle of second joint of the robot was exceeded, which was caused by saving not good enough “over the tray” pose. On iteration four and seven the motion supervision of the robot was triggered and stopped the robot movement. This scenario is presented in Figure 54, where the slope of the part makes the robot push the table when gripper fingers are approaching each other.

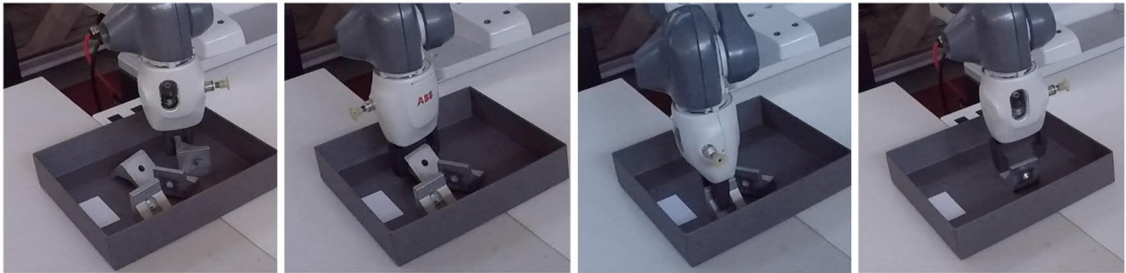


**Figure 54.** Slope of the part causing motion supervision error from the robot controller

From the five failed grasping, three of them was caused by the robot or environmental factors. Yet, with all of them the grasp pose itself was good. By considering these three as successful grasp poses, the accuracy of the grasp poses would be 95%, which is notably better. However, the Bin-Picking systems are evaluated as whole, meaning that

the successful grasping should be calculated instead of the successful grasp poses. The process itself doesn't have enough information for the user to save "good enough" poses, hence the task information given for the human should include the information about how the pose needs to be saved for the system to effectively use it.

In two iterations one of the robot fingers collided with the part. The Figure 55 illustrates how the fingers collided on the ninth iteration. First three grasp poses are fine, and the robot manages to bring the parts for the human, yet with the last part the right robot gripper finger collides with the part.



**Figure 55.** *Example of the grasp positions with a failed grasping in last image*

The grasp poses were additionally evaluated through the PCD presentation, which showed that the grasp poses in the successful grasping scenarios were correct. This would mean, that the calibration function worked, yet the find the grasp pose -function didn't in all cases. The point cloud was unclear in scenarios where the grasp pose was unsuccessful, and it seemed to be caused by the reflection of the part. The points near the collided finger were more infrequent and that's why the location of the grasp pose was enough on the left to make the right finger collide with the part. Therefore, the find grasp pose -function would require further development to overcome this issue.

## 7. CONCLUSIONS

The Thesis found working solution for the reusing the collaborative robot cell for Bin-Picking operations with the help of human operator. The solution consisted of modular context-aware Smart Factory components, which were able to seamlessly communicate with each other using FIWARE as middleware. The processes created to teach and evaluate the collaborative robot cell were successful and shaped the components to provide modular functions. However, the proposed solution is not an industrial grade and was created for research environment.

To conclude the work done in this Thesis, the developed solution and the results of the work are presented through the research questions provided in Sec. 1.2. The questions are answered in following paragraphs individually.

*Implementing gestural HRC based system for industrial applications* required research about HRI and HRC in general, and about previous work in the field of gesture recognition. The solution ensured safety of the operator through force and torque sensors and soft paddings installed to the YuMi robot, and with safe operating speeds of the robot. Furthermore, to exploit the flexibility of the human operator, the walk-through robot programming method was utilized to teach new poses for the robot, which were saved using hand gestures invoked by the operator.

For the gestures to be recognized M2O2P hand gesture recognition component was deployed, which provided high gesture recognition accuracy: 99.05% measured in the tests. Such accuracy is comparable what was reported in similar systems; In [124] the vision-based gesture recognition system uses ML algorithms and Kinect v2 to provide 98.9% accuracy, whereas in [125] the authors reported accuracy of 98.9% for sensor glove-based hand gesture recognition for Arabic sign language. M2O2P component was further evaluated by measuring the feasibility of the component in the means of accidental gestures. Testing with the Cutkosky's grasp taxonomy revealed four accidental gestures, which can be avoided when planning what gestures to use in tasks where object manipulation is necessary. Accidental gestures were not faced when doing the Use-Case tests.

For complete communication between the human operator and robot, there were Robot Controlling Application developed to provide the gestural actions triggered by human operator for the robot controller, and Orchestrator Application to assign tasks for all software components. Such components provided little to no variations within the process,

and therefore no errors or failures in functionality for such was reported in software runtime.

*Creating a component that processes 3D data for Bin-Picking operations* included research about the required information for successful Bin-Picking, such as poses of the object and the gripper, and necessary Point Cloud processing methods for finding such information. For this purpose, 3D-module was deployed with three functionalities: communicating with the 3D camera and acquiring a PCD, finding the grasp pose and calibrating the location of the camera using the acquired PCD. The latter one was added for the 3D-module to provide modularly attachable component for HRC system, which utilized chessboard tag located in known location for the robot controller and was used for finding the location of the camera with respect to the robot base frame. Furthermore, the calibration function was successful and worked on every iteration. However, problems regarding the shadow that the bin creates were faced and with the provided solution only way to fix it was to position the camera in a way, that the shadow is not cast on the calibration chessboard tag.

The algorithm created for 3D-module for finding the grasp pose assumed the parts consists mostly of 90-degree angles, are always picked from above and are semi-structured. The successful grasp poses were able to be found with a 95.0% accuracy and successful grasping of the system was reported as 87.5%. The developed system doesn't compete with the accuracies of other state-of-the-art systems, such as in [82] the reported accuracy was  $91.6 \pm 4.1\%$  with arbitrary objects without limitations for the shape of the object in random Bin-Picking setup.

*Gestural HRI method was compared to more traditional method using GUI* by creating a test, where the human operator taught the robot in three teaching iterations using the glove interface, and in three iterations using the mouse as interface. The time consumed completing the task and interacting with the system was analysed using video footage captured during the test. The test resulted with clear indication that for such teaching process the glove interface provided faster and more effortless interface to save the robot poses. On average, interaction using the glove consumed 29.7% from the whole human task, whereas the mouse took 41.2%.

## 7.1 Future work

M2O2P offers the support for one sensor glove, CaptoGlove. Due the software architecture of the component, with small changes in functions that process the sensor data to

states, the application supports any input device that can be utilized for gesture recognition. To further develop M2O2P, there could be more devices added that the application supports. In a case where the input device uses IMU sensors or vision sensors, the further development might require moving away from the static thresholds and utilize AI or ML for the solution.

The orchestrator assumes that the tasks are done in sequence, meaning there is no logic for handling parallel tasks. To make the OA a fully-fledged software, the ability to do the process modelling using GUI and the functions that supports multiple parallel tasks should be added. Furthermore, since Process Management (PM) as a topic is quite extensive, there should be further research done in this regard as a concept. It is acknowledged that in the Thesis the PM is not researched, since the area of research for Thesis would have been too large. Now the tasks that are given to RCA are executed one after other, and there is a small downtime between the tasks. In a real industrial environment OA and RCA should support trajectories with given ABB RAPID zone input. For example, the rest pose and the over the tray -pose could be given as a series of points and the trajectory would be smooth from point to another.

The solution finds the grasp pose for unknown parts by processing the point cloud automatically, which could be done considerably faster with ML/AI based approach. As mentioned earlier, 3D-module should support acquiring the PCD in the background, due the time it consumed in the process. The major limitations for the 3D-module were the shapes of the parts, consisting of mainly 90-degree angles, and missing support for random Bin-Picking. The support for different shapes and modifying the algorithm to support random Bin-Picking would be important for further development. The collision detection system assumes that the gripper fingers are fully opened when going in the grasp pose. If there is a part in between the grasp pose, and the function finds collision free position for the gripper, the system would accidentally pick two parts, which is not desired behaviour. The collision detection system should be further developed, by trying different distances between the gripper fingers and choosing it in a way that the gripper fingers are just outside of the pair of the points and the grasp pose is collision-free. Such change would mean that the gripper fingers should be controlled with the distance between the fingers instead of just opening and closing them.

## 8. REFERENCES

- [1] Lasi H, Fettke P, Kemper H-G, et al. Industry 4.0. *Business & Information Systems Engineering* 2014; 6: pp. 239–242.
- [2] Torn IAR, Vaneker THJ. Mass Personalization with Industry 4.0 by SMEs: a concept for collaborative networks. *Procedia Manufacturing* 2019; 28: pp. 135–141.
- [3] International Federation of Robotics. *Press Conference World Robotics 2021*, <https://ifr.org/ifr-press-releases/news/robot-sales-rise-again> (28 October 2021).
- [4] Schermerhorn P. DIARC: A Testbed for Natural Human-Robot Interaction. p. 8.
- [5] Chang W-C, Wu C-H. Eye-in-hand vision-based robotic bin-picking with active laser projection. *The International Journal of Advanced Manufacturing Technology* 2016; 85: pp. 2873–2885.
- [6] Truebenbach E. Is Automated Bin Picking finally real?, <https://www.universal-robots.com/blog/is-automated-bin-picking-finally-real/> (2019, accessed 14 August 2022).
- [7] Buchholz D. *Bin-Picking*. Cham: Springer International Publishing. Epub ahead of print 2016. DOI: 10.1007/978-3-319-26500-1.
- [8] Buchholz D, Winkelbach S, Wahl FM. RANSAM for Industrial Bin-Picking. pp. 1–6.
- [9] Buchholz D, Futterlieb M, Winkelbach S, et al. Efficient bin-picking and grasp planning based on depth data. In: *2013 IEEE International Conference on Robotics and Automation*. Karlsruhe, Germany: IEEE, pp. 3245–3250.
- [10] Shop4cf - SHOP4CF, <https://shop4cf.eu/> (accessed 9 April 2022).
- [11] Weiser M. The Computer for the 21st Century. p. 8.
- [12] Hozdić E. Smart Factory for Industry 4.0: A Review. *International Journal of Modern Manufacturing Technologies*.
- [13] Shi Z, Xie Y, Xue W, et al. Smart factory in Industry 4.0. *Systems Research and Behavioral Science* 2020; 37: pp. 607–617.
- [14] Lucke D, Constantinescu C, Westkämper E. Smart Factory - A Step towards the Next Generation of Manufacturing. In: Mitsubishi M, Ueda K, Kimura F (eds) *Manufacturing Systems and Technologies for the New Frontier*. London: Springer London, pp. 115–118.
- [15] Cyber-physical systems: Chancen und nutzen aus sicht der Automation. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik. 2013.
- [16] Xu LD, He W, Li S. Internet of Things in Industries: A Survey. *IEEE Transactions on Industrial Informatics* 2014; 10: pp. 2233–2243.



- [17] Goodrich MA, Schultz AC. Human-Robot Interaction: A Survey. *Foundations and Trends® in Human-Computer Interaction* 2007; 1: pp. 203–275.
- [18] Matheson E, Minto R, Zampieri EGG, et al. Human–Robot Collaboration in Manufacturing Applications: A Review. *Robotics* 2019; 8: p. 25.
- [19] Vysocky A, Novak P. Human-Robot Collaboration in Industry. *MM Science Journal* 2016; 2016: pp. 903–906.
- [20] Fong T, Thorpe C, Baur C. Collaboration, dialogue and human-robot interaction, 10th international symposium of robotics research (lorne, victoria, australia). In: *Proceedings of the 10th International Symposium of Robotics Research*. 2001.
- [21] Kofman J, Xianghai Wu, Luu TJ, et al. Teleoperation of a robot manipulator using a vision-based human-robot interface. *IEEE Transactions on Industrial Electronics* 2005; 52: pp. 1206–1219.
- [22] Fong T, Rochlis Zumbado J, Currie N, et al. Space Telerobotics: Unique Challenges to Human–Robot Collaboration in Space. *Reviews of Human Factors and Ergonomics* 2013; 9: pp. 6–56.
- [23] Luo RC, Wu Y-C. Hand gesture recognition for Human-Robot Interaction for service robot. In: *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. Hamburg, Germany: IEEE, pp. 318–323.
- [24] Hentout A, Aouache M, Maoudj A, et al. Human–robot interaction in industrial collaborative robotics: a literature review of the decade 2008–2017. *Advanced Robotics* 2019; 33: pp. 764–799.
- [25] Sheridan TB, Verplank WL. *Human and Computer Control of Undersea Teleoperators*: Fort Belvoir, VA: Defense Technical Information Center. Epub ahead of print 15 July 1978. DOI: 10.21236/ADA057655.
- [26] Aaltonen I, Salmi T, Marstio I. Refining levels of collaboration to support the design and evaluation of human-robot interaction in the manufacturing industry. *Procedia CIRP* 2018; 72: pp. 93–98.
- [27] Shi J, Jimmerson G, Pearson T, et al. Levels of human and robot collaboration for automotive manufacturing. In: *Proceedings of the Workshop on Performance Metrics for Intelligent Systems - PerMIS '12*. College Park, Maryland: ACM Press, p. 95.
- [28] Krüger J, Lien TK, Verl A. Cooperation of human and machines in assembly lines. *CIRP Annals* 2009; 58: pp. 628–646.
- [29] Huang C-M, Mutlu B. Anticipatory robot control for efficient human-robot collaboration. In: *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. Christchurch, New Zealand: IEEE, pp. 83–90.
- [30] Michalos G, Makris S, Tsarouchi P, et al. Design Considerations for Safe Human-robot Collaborative Workplaces. *Procedia CIRP* 2015; 37: pp. 248–253.
- [31] Asimov I. Runaround. *Astounding science fiction* 1942; 29: pp. 94–103.

- [32] SFS-EN ISO 10218-2 - Robots and Robotic Devices. Safety requirements for Industrial Robots. Part 2: Robot Systems and Integration (ISO 10218-2:2011). 5 September 2011.
- [33] ISO/TS 15066:2016(en), Robots and robotic devices — Collaborative robots, <https://www.iso.org/obp/ui/#iso:std:iso:ts:15066:ed-1:v1:en> (accessed 9 March 2022).
- [34] Lasota PA, Fong T, Shah JA. A Survey of Methods for Safe Human-Robot Interaction. *Foundations and Trends in Robotics* 2017; 5: pp. 261–349.
- [35] Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. St. Louis, MO, USA: Institute of Electrical and Electronics Engineers, pp. 500–505.
- [36] Flacco F, Kroger T, De Luca A, et al. A depth space approach to human-robot collision avoidance. In: *2012 IEEE International Conference on Robotics and Automation*. Saint Paul, MN: IEEE, pp. 338–345.
- [37] Kock S, Bredahl J, Eriksson PJ, et al. Better safety without higher fences. p. 4.
- [38] Vogel C, Walter C, Elkmann N. A projection-based sensor system for safe physical human-robot collaboration. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo, Japan: IEEE, pp. 5359–5364.
- [39] Suita K, Yamada Y, Tsuchida N, et al. A failure-to-safety ‘Kyozon’ system with simple contact detection and stop capabilities for safe human-autonomous robot coexistence. In: *Proceedings of 1995 IEEE International Conference on Robotics and Automation*. Nagoya, Japan: IEEE, pp. 3089–3096.
- [40] Takakura S, Murakami T, Ohnishi K. An approach to collision detection and recovery motion in industrial robot. In: *15th Annual Conference of IEEE Industrial Electronics Society*. Philadelphia, PA, USA: IEEE, pp. 421–426.
- [41] Wooten J, Bevly D, Hung J. Piezoelectric Polymer-Based Collision Detection Sensor for Robotic Applications. *Electronics* 2015; 4: pp. 204–220.
- [42] ABB. Collision Detection - RobotWare option, [https://library.e.abb.com/public/d4708d0240be2966c125772f00528ca9/Collision%20det%20PR10044EN\\_R2.pdf](https://library.e.abb.com/public/d4708d0240be2966c125772f00528ca9/Collision%20det%20PR10044EN_R2.pdf).
- [43] Sisbot EA, Marin-Urias LF, Broquère X, et al. Synthesizing Robot Motions Adapted to Human Presence: A Planning and Control Framework for Safe and Socially Acceptable Robot Motions. *International Journal of Social Robotics* 2010; 2: pp. 329–343.
- [44] Ryoo MS. Human activity prediction: Early recognition of ongoing activities from streaming videos. In: *2011 International Conference on Computer Vision*. Barcelona, Spain: IEEE, pp. 1036–1043.
- [45] Perez-D’Arpino C, Shah JA. Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA: IEEE, pp. 6175–6182.

- [46] Toichoa Eyam A, Mohammed WM, Martinez Lastra JL. Emotion-Driven Analysis and Control of Human-Robot Interactions in Collaborative Applications. *Sensors* 2021; 21: p. 21.
- [47] Yanco HA, Drury JL. A Taxonomy for Human-Robot Interaction. p. 10.
- [48] Gervasi R, Mastrogiacomo L, Franceschini F. A conceptual framework to evaluate human-robot collaboration. *The International Journal of Advanced Manufacturing Technology* 2020; 108: pp. 841–865.
- [49] Heimann O, Guhl J. Industrial Robot Programming Methods: A Scoping Review. In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vienna, Austria: IEEE, pp. 696–703.
- [50] Stolt A, Carlson FB, Ardakani MMG, et al. Sensorless friction-compensated passive lead-through programming for industrial robots. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany: IEEE, pp. 3530–3537.
- [51] Albu-Schaffer A, Hirzinger G. Cartesian impedance control techniques for torque controlled light-weight robots. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Washington, DC, USA: IEEE, pp. 657–663.
- [52] Calinon S, Billard A. Active Teaching in Robot Programming by Demonstration. In: *RO-MAN 2007 - The 16th IEEE International Symposium on Robot and Human Interactive Communication*. Jeju, South Korea: IEEE, pp. 702–707.
- [53] Mehrabian A. *Nonverbal Communication*. Routledge, 2017.
- [54] Dollar AM. Classifying Human Hand Use and the Activities of Daily Living. In: Balasubramanian R, Santos VJ (eds) *The Human Hand as an Inspiration for Robot Hand Development*. Cham: Springer International Publishing, pp. 201–216.
- [55] Spiliotopoulos D, Androutsopoulos I, Spyropoulos CD. Human-Robot Interaction based on Spoken Natural Language Dialogue. p. 5.
- [56] Rose RC, Paul DB. A hidden Markov model based keyword recognition system. In: *International Conference on Acoustics, Speech, and Signal Processing*. Albuquerque, NM, USA: IEEE, pp. 129–132.
- [57] Hinton G, Deng L, Yu D, et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine* 2012; 29: pp. 82–97.
- [58] Bingol MC, Aydogmus O. Performing predefined tasks using the human–robot interaction on speech recognition for an industrial robot. *Engineering Applications of Artificial Intelligence* 2020; 95: p. 13.
- [59] Schmitz A, Maiolino P, Maggiali M, et al. Methods and Technologies for the Implementation of Large-Scale Robot Tactile Sensors. *IEEE Transactions on Robotics* 2011; 27: pp. 389–400.
- [60] Salter T, Dautenhahn K, Boekhorst R te. Learning about natural human–robot interaction styles. *Robotics and Autonomous Systems* 2006; 54: pp. 127–134.

- [61] Yuan W, Dong S, Adelson E. GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force. *Sensors* 2017; 17: p. 21.
- [62] Atienza R, Zelinsky A. Active gaze tracking for human-robot interaction. In: *Proceedings. Fourth IEEE International Conference on Multimodal Interfaces*. Pittsburgh, PA, USA: IEEE Comput. Soc, pp. 261–266.
- [63] Chang H-T, Chang J-Y. Sensor Glove Based on Novel Inertial Sensor Fusion Control Algorithm for 3-D Real-Time Hand Gestures Measurements. *IEEE Transactions on Industrial Electronics* 2020; 67: pp. 658–666.
- [64] Coronado E, Villalobos J, Bruno B, et al. Gesture-based robot control: Design challenges and evaluation with humans. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. Singapore, Singapore: IEEE, pp. 2761–2767.
- [65] Dong W, Yang L, Fortino G. Stretchable Human Machine Interface Based on Smart Glove Embedded With PDMS-CB Strain Sensors. *IEEE Sensors Journal* 2020; 20: pp. 8073–8081.
- [66] Shen Z, Yi J, Li X, et al. A soft stretchable bending sensor and data glove applications. In: *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. Angkor Wat, Cambodia: IEEE, pp. 88–93.
- [67] Jones CL, Wang F, Morrison R, et al. Design and Development of the Cable Actuated Finger Exoskeleton for Hand Rehabilitation Following Stroke. *IEEE/ASME Transactions on Mechatronics* 2014; 19: pp. 131–140.
- [68] Lin B-S, Lee I-J, Yang S-Y, et al. Design of an Inertial-Sensor-Based Data Glove for Hand Function Evaluation. *Sensors* 2018; 18: p. 17.
- [69] Guo L, Lu Z, Yao L. Human-Machine Interaction Sensing Technology Based on Hand Gesture Recognition: A Review. *IEEE Transactions on Human-Machine Systems* 2021; 51: pp. 300–309.
- [70] Erol A, Bebis G, Nicolescu M, et al. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding* 2007; 108: pp. 52–73.
- [71] Yang H-D, Park A-Y, Lee S-W. Gesture Spotting and Recognition for Human-Robot Interaction. *IEEE Transactions on Robotics* 2007; 23: pp. 256–270.
- [72] Hwang T. Computational Power and the Social Impact of Artificial Intelligence. *SSRN Electronic Journal*. Epub ahead of print 2018. DOI: 10.2139/ssrn.3147971.
- [73] Kye Kyung Kim, Keun Chang Kwak, Su Young Ch. Gesture analysis for human-robot interaction. In: *2006 8th International Conference Advanced Communication Technology*. Phoenix Park, Korea: IEEE, p. 4 pp. – 1827.
- [74] Ellekilde L-P, Petersen HG. Motion planning efficient trajectories for industrial bin-picking. *The International Journal of Robotics Research* 2013; 32: pp. 991–1004.
- [75] Fujita M, Domae Y, Noda A, et al. What are the important technologies for bin picking? Technology analysis of robots in competitions based on a set of performance metrics. *Advanced Robotics* 2019; pp. 1–15.

- [76] ON Robot GR2-gripper - Flexible 2 finger robot gripper with wide stroke, <https://onrobot.com/us/products/rg2-gripper> (accessed 27 May 2022).
- [77] Robotiq vacuum gripper, [https://robotiq.com/products/vacuum-grippers?ref=nav\\_product\\_new\\_button](https://robotiq.com/products/vacuum-grippers?ref=nav_product_new_button) (accessed 27 May 2022).
- [78] ABB Single-arm YuMi IRB 14050 - Unchained Robotics, <https://unchainedrobotics.de/en/products/robot/cobot/single-arm-yumi-irb-14050/> (accessed 27 May 2022).
- [79] Gottschalk S, Lin MC, Manocha D. OBBTree: a hierarchical structure for rapid interference detection. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*. ACM Press, pp. 171–180.
- [80] Morales A, Sanz PJ, del Pobil AP, et al. Vision-based three-finger grasp synthesis constrained by hand geometry. *Robotics and Autonomous Systems* 2006; 54: pp. 496–512.
- [81] Jain A, Kemp CC. EL-E: an assistive mobile manipulator that autonomously fetches objects from flat surfaces. *Autonomous Robots* 2010; 28: pp. 45–64.
- [82] Klingbeil E, Rao D, Carpenter B, et al. Grasping with application to an autonomous checkout robot. In: *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, pp. 2837–2844.
- [83] Derpanis KG. Overview of the RANSAC Algorithm. p. 2.
- [84] Peng L, Zhao Y, Qu S, et al. Real Time and Robust 6D Pose Estimation of RGBD Data for Robotic Bin Picking. In: *2019 Chinese Automation Congress (CAC)*. Hangzhou, China: IEEE, pp. 5283–5288.
- [85] Ester M, Kriegel H-P, Xu X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. p. 6.
- [86] Trevor AJB, Gedikli S, Rusu RB, et al. Efficient Organized Point Cloud Segmentation with Connected Components. p. 6.
- [87] Dong Z, Liu S, Zhou T, et al. PPR-Net: Point-wise Pose Regression Network for Instance Segmentation and 6D Pose Estimation in Bin-picking Scenarios. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Macau, China: IEEE, pp. 1773–1780.
- [88] Qi CR, Yi L, Su H, et al. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. p. 10.
- [89] FAST-lab., <https://www.tuni.fi/en/research/fast-lab> (accessed 15 June 2022).
- [90] Caeiro-Rodríguez M, Otero-González I, Mikic-Fonte FA, et al. A Systematic Review of Commercial Smart Gloves: Current Status and Applications. *Sensors* 2021; 21: p. 33.
- [91] Product specification - IRB 14000. p. 110.
- [92] ABB. Product manual, IRB 14000 gripper.

- [93] Zivid One+ - Industrial 3D cameras for robot cells, <https://www.zivid.com/zivid-one-plus> (accessed 28 May 2022).
- [94] Zivid shop, <https://shop.zivid.com> (accessed 28 May 2022).
- [95] Docker, <https://www.docker.com/> (accessed 29 May 2022).
- [96] Docker volumes, <https://docs.docker.com/storage/volumes/> (accessed 29 May 2022).
- [97] FIWARE - About us, <https://www.fiware.org/about-us/> (accessed 28 May 2022).
- [98] Orion Context Broker - FIWARE, <https://fiware-orion.readthedocs.io/en/master/> (accessed 15 June 2022).
- [99] MongoDB, <https://www.mongodb.com/> (accessed 15 June 2022).
- [100] SHOP4CF - Data Models, <https://shop4cf.github.io/data-models/> (accessed 28 May 2022).
- [101] ETSI. Context Information Management (CIM); NGSI-LD API - ETSI GS CIM 009 V1.1.1.
- [102] SHOP4CF Data models, Task entity, <https://shop4cf.github.io/data-models/task.html> (accessed 1 June 2022).
- [103] SHOP4CF Data models, Device entity, <https://shop4cf.github.io/data-models/device.html> (accessed 6 January 2022).
- [104] ROS - Robot Operating System, <https://www.ros.org/> (accessed 29 May 2022).
- [105] ROS - Concepts, <http://wiki.ros.org/ROS/Concepts> (accessed 29 May 2022).
- [106] ROS 2 Documentation: Foxy, <https://docs.ros.org/en/foxy/index.html> (accessed 29 May 2022).
- [107] Zhou Q-Y, Park J, Koltun V. Open3D: A Modern Library for 3D Data Processing. 6.
- [108] CaptoGlove SDKs, <https://www.captoglove.com/sdks/> (accessed 19 June 2022).
- [109] ROS Wiki - rqt\_graph, [http://wiki.ros.org/rqt\\_graph](http://wiki.ros.org/rqt_graph) (accessed 29 May 2022).
- [110] Integration Service - eProsima, <https://github.com/eProsima/Integration-Service> (accessed 29 May 2022).
- [111] FIROS2 - eProsima, <https://github.com/eProsima/FIROS2> (accessed 29 May 2022).
- [112] Fonseca JMC, Márquez FG, Jacobs T. FIWARE-NGSI v2 Specification, <https://fiware.github.io/specifications/ngsiv2/stable/> (accessed 29 May 2022).
- [113] ABB. Technical reference manual, RAPID Instructions, Functions and Data types.

- [114] OpenCV Python, <https://github.com/opencv/opencv-python> (accessed 29 May 2022).
- [115] OpenCV Documentation - findChessboardCorners(), [https://docs.opencv.org/4.x/d9/d0c/group\\_\\_calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a](https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a) (accessed 29 May 2022).
- [116] Kabsch W. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A* 1976; 32: pp. 922–923.
- [117] Open3D, Tutorial, basic, Point cloud, <http://www.open3d.org/docs/0.8.0/tutorial/Basic/pointcloud.html> (accessed 29 May 2022).
- [118] Muja M, Lowe D. FLANN - Fast Library for Approximate Nearest Neighbors, User Manual.
- [119] Open3D, point cloud outlier removal, Statistical outlier removal, [http://www.open3d.org/docs/latest/tutorial/Advanced/pointcloud\\_outlier\\_removal.html](http://www.open3d.org/docs/latest/tutorial/Advanced/pointcloud_outlier_removal.html) (accessed 1 June 2022).
- [120] Edelsbrunner H. Alpha Shapes - A Survey. *Wiadomości Matematyczne* 2012; 48: p. 47.
- [121] Delaunay B. Sur la sphère vide. A la mémoire de Georges Voronoï. *Bulletin de l'Académie des Sciences de l'URSS Classe des sciences mathématiques et na* 1934; pp. 793–800.
- [122] QHull Library, <http://www.qhull.org/> (accessed 1 June 2022).
- [123] Cutkosky MR. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Transactions on Robotics and Automation* 1989; 5: pp. 269–279.
- [124] Mazhar O, Navarro B, Ramdani S, et al. A real-time human-robot interaction framework with robust background invariant hand gesture detection. *Robotics and Computer-Integrated Manufacturing* 2019; 60: pp. 34–48.
- [125] Tubaiz N, Shanableh T, Assaleh K. Glove-Based Continuous Arabic Sign Language Recognition in User-Dependent Mode. *IEEE Transactions on Human-Machine Systems* 2015; 45: pp. 526–533.