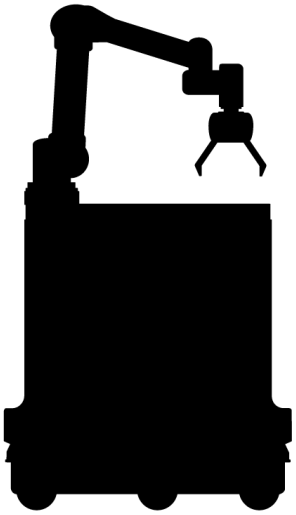
Special Priority Order in an Industry 4.0 Facility with Little Helper 6

- Robots in an Application Context -



Bachelor Report

17GR664

Aalborg University
Robotics

Robotics
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Special Priority Order in an Industry 4.0 Facility with Little Helper 6

Theme:
Robots in an Application Context

Project Period:
6th Semester Spring 2017

Project Group:
17gr664

Participant(s):
Biranavan Pulendralingam
Emil Blixt Hansen
Rasmus Andersen
Steffen Madsen

Supervisor(s):
Rasmus Skovgaard Andersen
Rikke Gade

Copies: 3

Page Numbers: 92

Date of Completion:
May 30, 2017

Abstract:

The motivation for this project is to automate logistic processes and part feeding tasks related to customised orders in an Industry 4.0 environment. This is achieved by utilising the Little Helper 6 (LH6), which is the latest model from a series of Autonomous Industrial Mobile Manipulators (AIMM), developed by Aalborg University. The environment is set in an Industry 4.0 production line demo, assembling smart-phones dummies. In order for the LH6 to successfully perform specialised custom orders, a communication between LH6 and the environment has been established, by making the LH6 a resource in the MES of the production demo. To enable LH6 to locate, pick and place the desired smart-phone covers, a vision system has been made. The same vision system enables LH6 to calibrate the manipulator with respect to the static environment using a QR-code. Due to communication error and manipulator imprecision, a success rate of 60% was achieved in the final test. The solution did not fulfil all the requirements, but shows potential for future industrial solutions.

Contents

Preface

This report documents a project made by 6th semester students of Robotics B.Sc. at Aalborg University. The project was carried out during the time from February to June 2017. The complexity of technical and programming language is on a level, that is expected of students on this education program to use and understand. The topic of this project is specialised custom orders in an Industry 4.0 environment.

A special thanks to Rasmus Skovgaard Andersen and Rikke Gade, the project supervisors, who helped the authors and encouraged them in the process of writing the report and developing the solution.

Reader's Guide

This report uses the modified Chicago style of reference, formatted as (author's last name, year). All references used, are located in the end of the report. For figures, references is located in the caption. If no reference is present, it means that the figures were created by the authors themselves.
In Appendix A a setup guide of LH6 and FESTO CP Factory is given.
All additional material can be found in Google Drive:
`https://drive.google.com/open?id=0BzGSI-iEysv6US1kMUJ3SHRFZDA`
or Figure 1.

(a) Additional material.

(b) Demonstration video.

Figure 1: QR-codes to additional material and demonstration video.

Aalborg University, May 30, 2017

Biranavan Pulendralingam
<bpulen14@student.aau.dk>

Emil Blixt Hansen
<ebha14@student.aau.dk>

Rasmus Andersen
<rean14@student.aau.dk>

Steffen Madsen
<smad14@student.aau.dk>

Chapter 1

Introduction

The goal of this project is to automate logistic processes and part feeding tasks related to customised orders in an Industry 4.0 environment. This is achieved by creating a vision system for an Autonomous Industrial Mobile Manipulator (AIMM), called Little Helper 6 (LH6) and establish communication between the LH6 and an Industry 4.0 demo located at Aalborg University.

This chapter consists of an introduction to the project context, a description of the LH6, the FESTO CP Factory and an initial problem statement.

1.1 Project Background

As a result of an expanded globalised market, an increased demand for customisation and low product life cycles, manufacturing companies are currently experiencing a paradigm shift within manufacturing towards mass customisation. The impact of this, is that modern production systems should be able to handle; small product life cycles, small batch sizes and a big product variation. If a production system fails to meet these demands, it can be a threat for the competitiveness and the survival of the manufacturing company. Therefore, in order for a manufacturing company to stay competitive, they need an efficient, flexible and reconfigurable production system, with the ability to execute according to customised orders.

The current marked for robot-based manufacturing, is an essential part of many highly automated production systems. These systems often offers efficiency due to statically fixed industrial robots, placed in a cell, doing predefined actions, but lacks when it comes to flexibility and reconfigurability. Alternatively, manual labour brings a great flexibility, but is often not economically viable. (Madsen 2016) (Pedersen et al. 2016)

In the annual road map developed by SPARC, an EU funded agent for robotic strategies in Europe, the increasing need for robots within a span of different domains, e.g. robot based manufacturing, healthcare and agriculture are recognised. In the road map SPARC identifies the state of art and future challenges for essential robotic abilities, some of which are listed in the following:

Configurability:
The ability of a robot to configure to different setups and a range of different tasks, by plug and play architectures. This ability applies on configuration of software, electronic modules and the mechanical system of a robot. SPARC identifies the key challenges for this ability to be the need for standardised interfaces in plug and play architectures.

Adaptability:
The ability of a robot to adapt to different environments, work scenarios and inputs. This ability applies on adoption of sensor processing, control changes in actuators, adoption of new environments and long term self calibration on intrinsic and extrinsic calibration parameters.

Interaction ability:
The ability of a robot to interact with systems, users, operators or other robots. This ability applies for a wide area of interaction, spanning from simple communication protocols to interactive social communication. A key challenges for this ability is the user acceptability for interactive robots and the certification and validation required for safety critical task.

Motion ability:
The ability of a robot to move, with precision and repeatability in different environments and on different surfaces. The identified key challenges for this ability is technological limitations.

Perception ability:
The ability of a robot to perceive its environment. Simplified, it applies on the probability of accurately detecting objects, spaces and locations using robot perception. The key challenges of this ability, is the lack of generic off-the-shelf and affordable sensor technologies for detecting and sensing specific materials, e.g. reflective or transparent materials.

Decisional autonomy:
The ability of a robot to operate autonomously, spanning from a simple RFID sensor stopping a conveyor in a production line to a self-operating robot in complex and dynamic environments. The key challenges are the lack of sophisticated solutions to complex robot decision making in uncertain environments.

Within the manufacturing domain SPARC explicitly recognises the need for a flexible and reconfigurable production together with the need for a bigger impact range of robot-based manufacturing. According to SPARC, some the expected robotic abilities enabling this is autonomous robotic solutions with intuitive on-the-fly programming and the ability to interact with humans, in a dynamic and little known environment. (Europe 2017)

1.2 Little Helper

The pursuit for a flexible and reconfigurable production and an investigation of the key challenges proposed by SPARC, is a field which has drawn a great attention from researchers at Aalborg University, which among other has resulted in a family of six AIMMs, called Little Helpers (LH). LH is Aalborg University's idea of a low cost element in an efficient, flexible, reconfigurable and highly automated production system. By an integration of off-the-shelf grippers, manipulators and mobile platforms, a LH is able to move autonomously and solve several industrial tasks such as pick and place and part feeding. On-the-fly programming of LH's is achieved through a skill based program called Skill Based System (SBS). For this project the newest AIMM member (LH6) will be used (see Figure 1.1).

Figure 1.1: Little Helper 6.

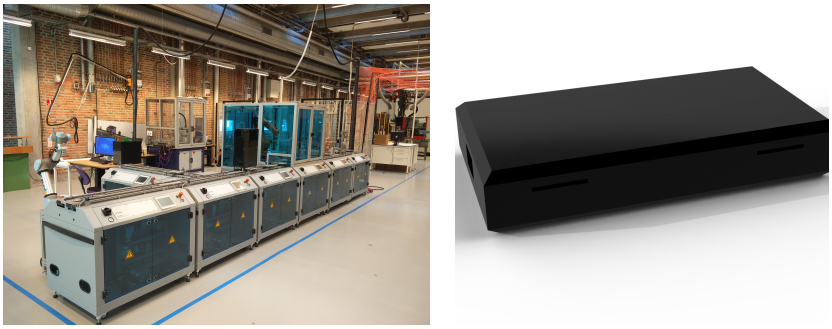
1.3 Industry 4.0

Along with the increased need for flexibility and reconfigurability within a production system a new digital industrial trend has emerged, called Industry 4.0. It combines new technologies within internet, robotics and mechanics. By connecting robots, machines and sensors through a network, Industry 4.0 makes it possible to gather and analyse data across a production. In terms of Industry 4.0 there are

nine major emerging technologies e.g. autonomous robots, simulation, the industrial internet of things and cloud computing.

According to a report developed by the global management consulting firm in business strategies, the Boston Consulting Group (BCG), Industry 4.0 will transform nowadays manufacturing. The BCG report states that Industry 4.0 will cause a shift from single automated cells, to fully automated and integrated production systems, with an increased productivity, flexibility and quality. Researches done by BCG have shown that Industry 4.0, can create up to 390,000 jobs, and increase the manufacturing investments with € 250 billion alone in Germany. (Russmann et al. 2015)

At Aalborg University, an Industry 4.0 demo is implemented, referred to as FESTO CP Factory, with the aim of combining the nine emerging technologies of Industry 4.0 into one production system. FESTO CP Factory produces black dummy smart-phone (see Figure 1.2b) and consists of several cypher physical systems as shown in Figure 1.2a. All system are independent and connected through a network, which makes FESTO CP Factory in whole reconfigurable.



(a) FESTO CP Factory. **(b)** Black dummy smart-phone.

Figure 1.2: FESTO CP Factory and black dummy smart-phones.

1.4 Initial Problem Formulation

The aim of this project is to automate part feeding and logistic related to customised orders at the FESTO CP Factory, and thus increasing the flexibility, reconfigurability and the product variation in a FESTO CP Factory.

How can the product variety at FESTO CP Factory be expanded utilising LH6?

Chapter 2

System Description and Analysis

This chapter will give a description of FESTO CP Factory. This is done in order to get an overview of how the cypher physical systems at FESTO CP Factory are connected, so that smart-phone customisation in the production can be increased with the help of an AIMM.

2.1 FESTO CP Factory

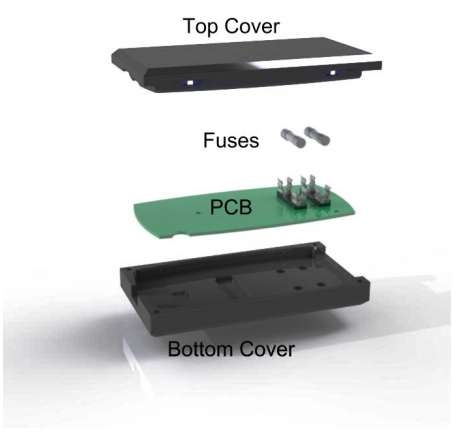
FESTO CP Factory, consisting of eight modules, serves as a demo for assembling black dummy smart-phones. These smart-phones consists of a top and bottom cover, up to two fuses, two or four holes and a PCB (as seen in Figure 2.1a). The modules of FESTO CP Factory are independent, meaning they can be programmed to do individual tasks and be rearranged without making any changes to what they are programmed to produce. Each module has a conveyor used to transport carriers between the modules. On each carrier a read-/writeable RFID tag is located enabling a flexible environment (Figure 2.1b). When a module reads an RFID tag, it asks the Manufacturing Execution System (MES) for what the modules should do with the part on the current carrier.

2.1.1 Assembly Flow

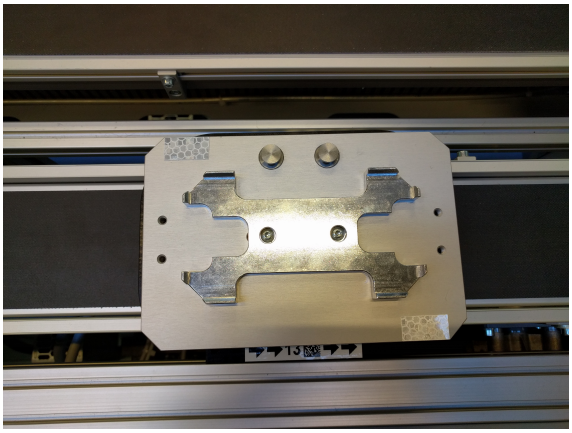
In Figure 2.2 the current configuration of FESTO CP Factory is shown. The circle numbers describes the flow of the assembly.

1) Bottom cover dispenser
The flow starts at the bottom cover dispenser. Here the dispenser will place a bottom cover on a carrier. The orientation does not matter as long as the bottom cover is placed on its back and correctly fits the carrier.

2) Drill machine
Here FESTO will drill holes in the bottom cover.



(a) The parts of the dummy smartphone.



(b) Empty carrier running on FESTO CP Factory.

Figure 2.1

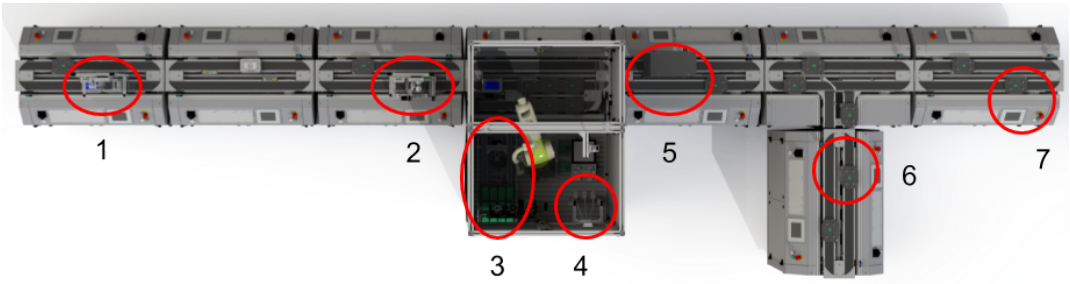


Figure 2.2: Current configuration of FESTO CP Factory.

3) PCB assembly

Entering this module containing a KUKA manipulator, the carrier will go onto a side conveyor, so carriers not supposed to have work done by the KUKA manipulator, can pass. The KUKA manipulator places the cover on a backlit plate to check the orientation. It will then pickup a PCB and place it on the bottom cover.

4) Fuse dispenser

The KUKA manipulator picks one or two fuses and places them on the PCB. The KUKA manipulator will then move the bottom cover, containing the PCB and fuses, back to its carrier. The carrier will be passed on and merged in to the main conveyor.

5) Fuses quality control

To ensure that the fuses are correctly placed on the PCB, an image quality control is performed.

6) Top cover dispenser
A station is dedicated for a feeder to place the top cover. Note, that here the orientation of the top cover is important.

7) Manual pick up
At the end of a production, a module is dedicated for manually picking of the finished smart-phone.

The total time it takes for all of these processes is 4 minutes and 3 seconds. If the carrier is doing a round on the FESTO CP Factory without doing any operations, it will take 2 minutes and 20 seconds.

2.1.2 Manufacturing Execution System

MES runs on a single computer where all orders are placed. It is in MES that a customer customise what he/she wants for the dummy smart-phone e.g. number of fuses and holes. When an order is placed, MES saves the recipe for the order in a database on the computer. This recipe contains which modules needs to be visited to produce the given order. For a standard order of one phone with two holes, and 1 fuse, the recipe will be; cover dispenser module $\rightarrow$ drilling module $\rightarrow$ PCB assembly $\rightarrow$ fuse dispenser module $\rightarrow$ quality check module $\rightarrow$ top cover dispenser module $\rightarrow$ manual pick up module. All orders are assigned to its own carrier and thus an RFID tag.
Every time a module reads an RFID tag and asks MES what to do, MES will look up that RFID tag for its recipe in the database. If the RFID tag corresponds with the next module in the recipe, MES will tell the module to perform its action (e.g. drilling). As seen in Figure 2.3, all communication between MES and all the PLC's happens through a network, in this case a TCP/IP connection.

2.1.3 System Limitations

In the current system it is only possible to order black phone covers. MES supports adding new modules, so it is possible to add multiple dispenser modules each with a separate colour. This, however, is not necessarily desirable since one phone cover colour can, for various reasons, be ordered less frequent than others. This means that the modules with the less-frequent ordered colours will be idle for a majority of the time, which is not desirable.

2.2 Little Helper 6

As mentioned in Chapter 1 LH6 is the sixth version of the LH family. LH6 consists of four main parts, Robotiq 3-Finger Gripper, UR5, mounting table and a MiR100. These parts are shown in Figure 2.4. (Andersen et al. 2016)

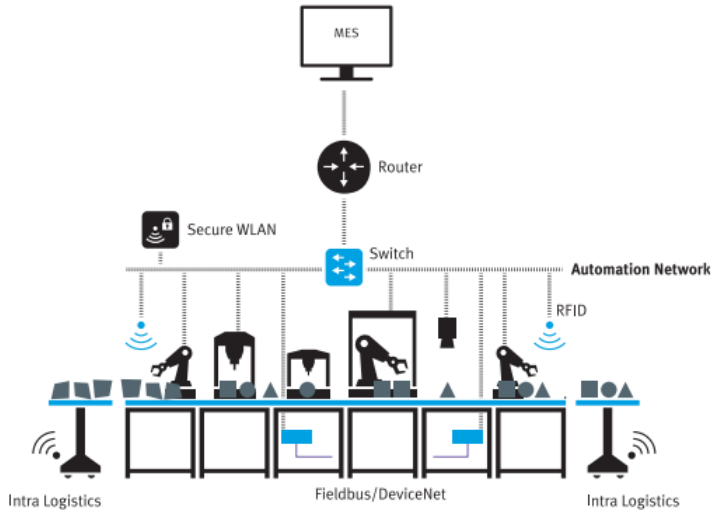


Figure 2.3: Overview of network connection in a CP Factory. (Festo-Didactic-SE 2017)

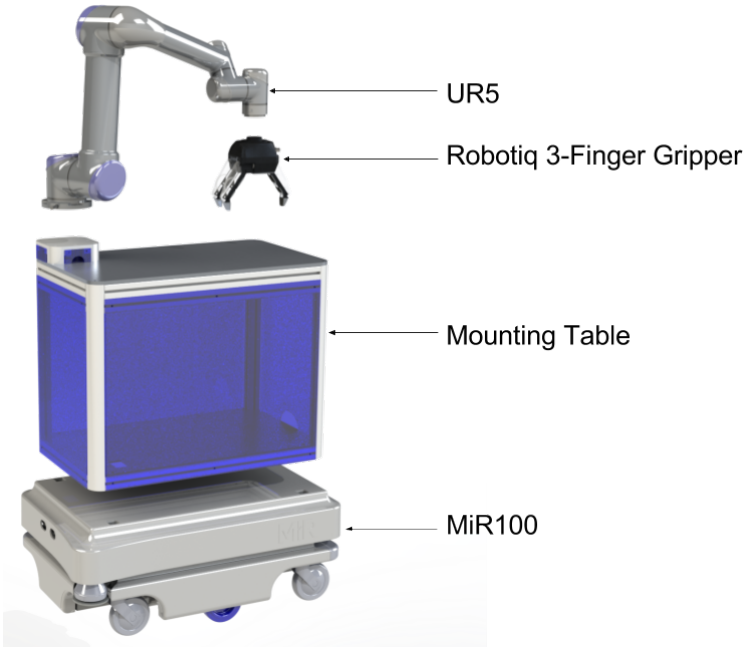


Figure 2.4: The four main parts of LH6.

2.2.1 Specification of LH6

Since three of these four parts are off-the-shelf commercial products (Robotiq gripper, UR5, MiR100), the specifications of LH6 are related to these products. The MiR100 has a precision of $\pm$ 10 cm, and the UR5 has 6 degrees of freedom, a pay-

load of 5 kg and a repeatability of $\pm$ 0.1 mm. The Robotic 3-Finger Gripper has a payload of 10 kg and weights 2.3 kg. The combined specifications of LH6 are shown in Table 2.1. Due to the mobile precision of the LH6, interactions with the environment generally requires position calibrations after each mobile movement e.g. using camera systems. (Andersen et al. 2016)

Description	**Value**	**Unite**
Platform precision	$\pm$ 10	cm
Manipulator repeatability	$\pm$ 0.1	mm
Manipulator payload	2.7	kg
Manipulator reach	850	mm
LH6 payload	20	kg
Tabletop height	975	mm
Max height in home position	1595	mm
Software	SBS	-
Price	491 200.91	DKK

Table 2.1: Specification of LH6. (Andersen et al. 2016)

2.2.2 Skill Based System

Skill Based System (SBS) is a software framework developed for the LH family. It is an intuitive framework developed for easy programming of AIMM's for non-robot experts. SBS is build upon the open source Robotic Operation System (ROS) and is programmed in C++ and Python. Its user interface (UI) is developed through a software platform by Qt Company. SBS is build on the concep of a hierarchy shown in Figure 2.5. (Pedersen et al. 2016) These three levels are described as:

Device primitives: The lowest level of commands in SBS e.g. simple movements of manipulator, opening of gripper ect.

Skills: Hence the name, skills are the foundation of SBS and are intuitive object centred robot abilities. A skill can consist of several Device Primitives depending on the complexity of the given skill. A skill could e.g. be a pick skill, where the user teaches the robot where the object is located and how to grasp it. The teaching is done by manually moving the robot around. In Figure 2.6 it is illustrated that a skills is build up of certain conditions. The first part of a skill is to check if certain preconditions are matched, e.g. check if the gripper is open before performing a pick skill. After a successful precondition check, the execution of the skill is carried out. After the execution, a check is performed to investigate if the prediction match-up with the result of the execution (also called postcondition).

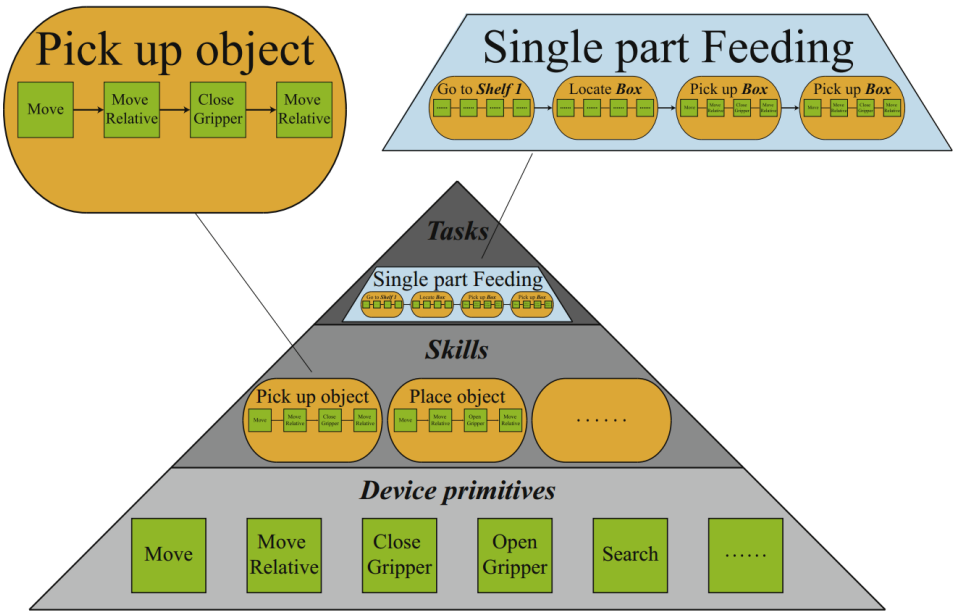


Figure 2.5: The hierarchy of SBS. (Pedersen et al. 2016)

Tasks: The top layer of the hierarchy is Tasks. A Task consists of one or more skills, which enables the robot to do a whole task, e.g. driving to a location performing a pick and place and drive away again.

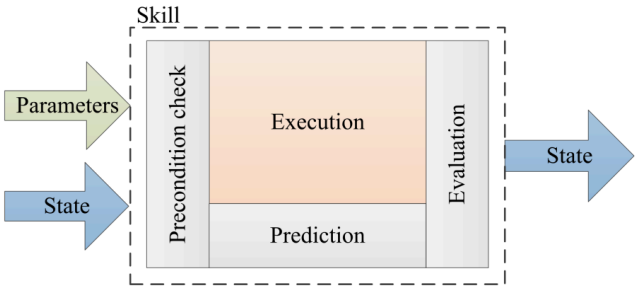


Figure 2.6: A model of the skill structure. (Bøgh et al. 2012)

2.3 Use Case Description

This project will focus on locating blue and white dummy smart-phones located in a storage facility (as seen in Figure 2.7a) and feeding them to the FESTO CP Factory, and thus automating special custom orders. The black, blue and white dummy smart-phones are identical to each other besides their colours. In Figure 2.7b the three dummy smart-phones are shown in complete assembly.

(a) The storage facility. **(b)** All three fully assembled phone covers.

Figure 2.7: The three dummy smart-phones and the storage facility.

Because the three covers are identical besides their colours all operations mentioned in Section 2.1 are the same except the "Bottom cover dispenser" and "Top cover dispenser" which will be replaced by the LH6.
The blue and white smart-phone dummies are located on a table in the storage facility. On this table both top and bottom covers of both blue and white covers are located as seen in Figure 2.7a. The phone covers are randomly placed and oriented with 3 degrees of freedom, i.e. they are all placed on their back facing upwards and can rotated around their z-axis. This means that perception ability for the LH6 is required in order to locate the randomly placed covers. Furthermore, communication between the MES and the LH6 is needed for the LH6 to know when to drive to the storage facility.
The complete use case is illustrated in Figure 2.8.

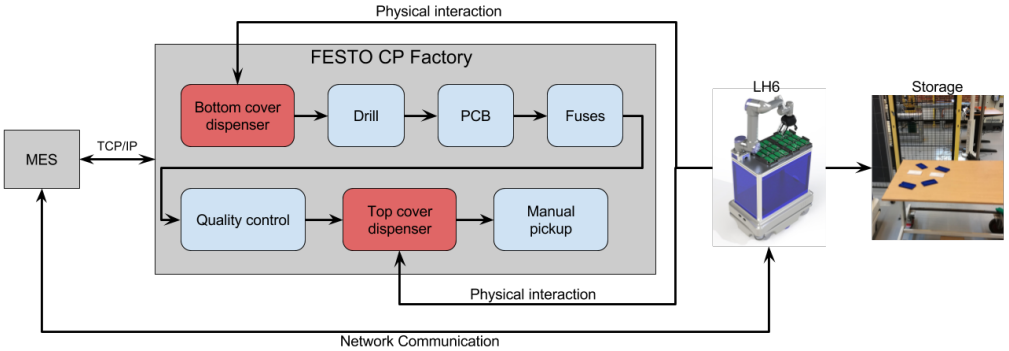


Figure 2.8: Flow of actions on FESTO CP Factory, where the operation that are colour depended, are coloured red.

Chapter 3

Image Processing Challenges

By adding vision to a robot solution the possibility to use physical entities instead of 3D coordinates during robot programming can be achieved, simplifying the human-robot interaction. Furthermore, vision can be used to execute pre- and postconditions of skills which are two important aspects of object centred robot abilities (see Section 2.2.2). (Pedersen et al. 2016)
This chapter will look into the traditional steps of image processing and examples of visions systems from earlier LH's.

3.1 Image Processing

Traditional image processing can be divided up into 5 steps, as illustrated in Figure 3.1). (Moeslund 2012)

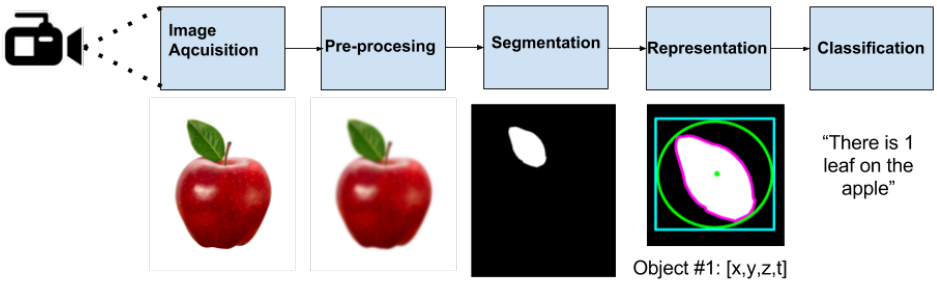


Figure 3.1: The traditional steps of image processing.

Image acquisition
By carefully selecting the camera setup for a project, the quality of the image can be controlled and noise can be reduced. This decreases the complexity of the following processes.

Pre-processing
By manipulating the image, features in the image such as shapes or colours can be enhanced while the noise-level is decreased or removed.

Segmentation
The aim of segmentation is to divide an image into different segments, where each segment is a regions of interest. Thus the segments represent the foreground, and the rest is the background. Segmentation requires robustness against e.g. changes of viewpoint and illumination. If the segmentation algorithm is not robust, small changes to the environment can make the segmentation fail. When the segmentation is done, the output is be a binary image with Binary Large Objects (BLOB) where segments are marked with 1 (white) and background is marked with 0 (black).

Representation
In order to compare objects in an image, different features, such as area and/or edge lengths, can be extracted from the regions of interest. This means that each region is now represented by a list of features describing the physical appearance of the object. The features from one object can then directly be compared with the same features from another object.

Classifications
When each object is represented by different features, they can be classified. The classification is often done by using a classifier and a set of training data. Whenever a new object is to be classified, its features are compared to the training data. The new object is then classified as the same class as the set of training data which it shares the most similarity with, e.g. by comparing the euclidean distance between the new features and the training data features.

3.2 Existing LH Vision Systems

Vision systems has been made for previous LH's and the following list are examples of such systems, giving a short description of what they were used for in an industrial context. The vision systems for the LH's are divided into three subjects; Object Detection and Position Estimation, Quality Control and Position Calibration where each of these has been integrated into the SBS framework.

Object detection and pose estimation
The paper (Andersen et al. 2013b) gives an example of how a vision system can be used to locate objects in order to pick them up. The system used methods such as edge detection and ray projection from the camera down to the object, in order to find the location and pose of the object, using a camera mounted on the gripper.

Quality control

In the paper (Andersen et al. 2013b) the goal was to detect if a magnet had been placed correctly in a rotor. The offset would be measured according to the distance and angle between a magnet and a rotor core, and compared to a pre-specified thresholds as shown in Figure 3.2.

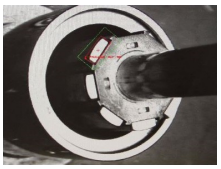


Figure 3.2: Quality control of magnet. (Andersen et al. 2013b)

Position calibration

In the paper (Andersen et al. 2014) an alternative method for calibrating the position of an AIMM is proposed. The paper introduces fast calibration by using a 3D camera and a QR-code. The QR-code acts as a static reference in the world and a pointcloud from the 3D camera is used to estimate a transformation from the QR-code to the camera. As shown in Table 3.1 the paper concludes the proposed method is at least 10 times faster than traditional alternatives with the downside of being less precise.

Method	Duration	Precision	Source
Haptic	30-45 sec	$\pm$ 1.0 mm	(Pedersen 2011)
High speed	10 sec	$\pm$ 1.0 mm	(Hvilshøj et al. 2010)
High Precision	60 sec	$\pm$ 0.1 mm	(Hvilshøj et al. 2010)
Proposed method	< 1 sec	< $\pm$ 4.0 mm	(Andersen et al. 2013a)

Table 3.1: Calibration method speed.

Chapter 4

Problem Formulation

The demands of manufacturing is trending towards higher flexibility in order to accompany increased customer demands for customisation, as mentioned in Chapter 1.
The goal of this project is to increase the product variety in FESTO CP factory, by the use of an AIMM, namely the LH6, using a vision system. The increased product variety, includes blue and white dummy smart-phones. These are viewed as a special priority product with low occurrence. FESTO CP Factory produces black dummy smart-phones, thus the expansion of blue and white is needed to be integrated. The environment is set in a non-enclosed workshop with people present where LH6 should be able to move around. A vision system is needed both at the FESTO CP Factory and at the storage facility for fetching locating randomly placed parts. Previous research done using a LH robot has shown that a implementation of vision systems for industrial task, can be successfully implemented in such environment. Furthermore, a communication between LH6 and FESTO CP Factory is needed for successful implementation.
Final problem statement: *How can blue and white smart-phone dummies be integrated into the FESTO CP Factory using LH6 with a vision system and SBS?*

Chapter 5

Requirement Specification

This chapter will give a description of the requirements for this project along with the delimited case and solution proposal.

5.1 Requirements

The requirements are split into three parts; *hardware*, *software* and *operation*. The *hardware* requirements listed includes the camera, lens and network setup where *software* includes network communication, localisation stability and pose estimation and *operation* includes operation time and delay.

5.1.1 Hardware

H.1: Camera position
When the camera is used to locate covers, it should be able to detect covers with a different height and angle to the table beneath, while the camera being mounted on the gripper.

H.2: Camera
The camera used should be RGB and have a resolution of atleast full HD (1920x1080) to ensure that enough details are captured by the camera.

H.3: Field of view
The lens should allow a field of view so that the image plane will represent at least the size of the carrier, which has the dimensions 18x13 cm.

5.1.2 Software

S.1: Software framework
The camera pose estimation software used to detect phone covers should be interfaced with SBS.

S.2: User friendliness UI
The SBS implementations in this project should have an easy to use interface.

S.3: Localisation time
To match the calibration speed of the 3D calibration method mentioned in Section 3.2, it should take less than one second to detect and pose estimate a cover.

S.4: Pose estimation precision
The carriers are designed with a funnel, allowing phone covers to slide into place. Thus it is estimate that the precision of the pose estimation should be within $\pm$ 5 mm and $\pm$ 5 degrees.

S.5: Interface between MES and SBS
For easy communication between MES, on FESTO CP Factory, and SBS on LH6, an interface between MES and SBS is needed.

S.6: Interface between MES and SBS should be generic
The interface between MES and SBS should be generic, meaning that it should be easy to chose what information is needed from MES, for a given application.

S.7: Image acquisition setup
The image acquisition setup should be able to compensate for change in light at the storage facility and at FESTO CP Factory.

S.8: Cover classification
The classification algorithm should have success rate of at least 99%, within normal working hours (8-16 o'clock).

S.9: Top cover orientation
The orientation of a top cover should be known, when it is picked up from the storage. Therefore, a method which is able to secure the cover orientation at least 99% of the time is required.

5.1.3 Operation

O.1: FESTO CP Factory delay
To ensure the most optimal operation time of FESTO CP Factory, the system must not wait for LH6 to perform actions.

O.2: Success rate of 95%
To ensure the stability of the system, it is required that it the implementation works 95% of the time.

O.3: Storage facility
The storage facility should have a static background in order to reduce noise doing image processing.

O.4: Operation time
It takes approximately 2 minutes and 20 second for a carrier to do a normal lap (without going to the "branch") on the FESTO CP Factory. Since this is a special order where an AIMM is involved, it is desired that no more than two laps (4 minutes and 40 seconds) is needed to complete the production cycle.

O.5: Error handling
The system should be able to handle a situation, where it is not able to fetch a part.

O.6: Safety
For LH6 to work in a factory without being placed in cell, it should follow the ISO 10218-1 and ISO 10218-2. (World 2016)

O.7: Storage facility placement
In order to not decrease the efficiency of FESTO CP Factory the storage facility can be placed maximum 10 m away from the nearest FESTO module.

O.8: LH6 placement
LH6 must not at any time be further away from FESTO CP Factory and the storage facility than 10 m. This is to ensure that a good WiFi connection is maintained.

5.2 Delimitation

For the scope of this project, delimitations are made by excluding requirements specified in Section 5.1. The excluded requirements are shown in Table 5.1. Since the storage facility and the placement of FESTO CP Factory is in a workshop with windows in the celling, the light will differ over the course of a day. Thus, the project is delimited to only work during day hours; more specific from 8 to 16 o'clock, thus requirement **S.7** is withdrawn from the final list of requirements.

The aim of this project is only to enable the system to assemble priority blue and white dummy smart-phones. Because of this and the fact that SBS is already an established program, the project focus will not include the user experience and its UI, removing requirement **S.2**.

A certain module on the FESTO CP Factory was chosen to be the one where all physical operations between LH6 and FESTO CP Factory will be carried out. This

	Description	Value	Unit
S.2	User friendly UI	-	-
S.6	Generic interface	-	-
S.7	Image acquisition compensate for light change	-	-
O.1	Delaying FESTO CP Factory	-	-
O.6	Safety	-	-

Table 5.1: Table of excluded requirements.

module is shown in Figure 5.1. Since one of FESTO CP Factory's main purpose, is easy re-arrangement of the modules, the layout may change. Nonetheless, this should not affect the use of the chosen module, only transportation time for LH6 between FESTO and storage facility.

Figure 5.1: The chosen module on FESTO CP Factory is highlighted.

Since this project serves as a proof of concept, requirement **O.1** is removed. Furthermore, it is chosen that it is acceptable to do the complete production within a timespan of 4 laps (9 minutes and 20 seconds), therefore requirement **O.4** is increased to 560 sec.

Due to the time constrains of this project, a full generic interface between MES and LH6 will not be investigated deeply, and requirement **S.6** is omitted for the final requirements. Furthermore, the safety of the operation will not be investigated since the environment is located in a lab on concept level, thus requirement **O.6** is removed.

Requirement **H.1** is changed because of time constrains, meaning that the height of the camera should be static and approx. perpendicular to the table. The height from the module to the camera, and the storage table to the camera should be approx. 445 mm.

The mentioned delimitations are done because of time and resource constrains. With these delimitations taken into consideration, a list of final requirements are established in Section 5.3.

5.3 Final Requirements

The following section consist of the delimited requirements. All the requirements left out from Section 5.1, is reserved for future work. The delimited requirement specifications are shown in Table 5.2.

	Description	**Value**	**Unit**
H.1	Static camera position above table	-	-
H.1	Camera view angle with table	90	degrees
H.2	Camera is RGB and minimum resolution	1920x1080	pixels
H.3	Minimum field of view	32.38	degrees
S.1	Software framework	SBS	-
S.3	Localisation time	1	s
S.4	Pose position estimation precision	$\pm$ 5	mm
S.4	Pose angle estimation precision	$\pm$ 5	degrees
S.5	Communication interface between MES and SBS	-	-
S.8	Cover classification	99	%
S.9	Top cover orientation	99	%
O.2	Stability of implementation	95	%
O.3	Storage facility with static background	-	-
O.4	Operation time for complete production cycle	560	seconds
O.5	Error handling for missing part	-	-
O.7	Storage facility maximum distance to FESTO	10	m
O.8	LH6 placement distance	10	m

Table 5.2: Delimited requirements.

s

5.4 Solution Proposal for Delimited Case

Since the project has been delimited in Section 5.2 and a set of requirement specifications (Section 5.3) has been made, a general solution proposal can be made. This section will give a step by step general description of the delimited case.

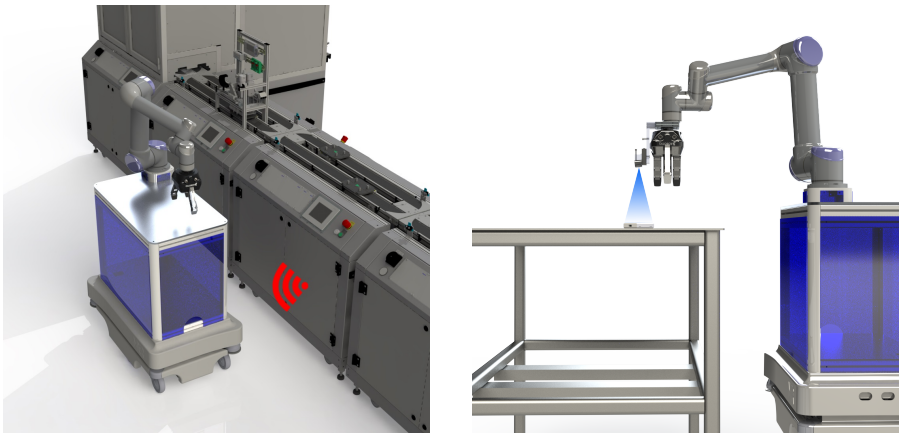
Order placement: When a user places a special priority order (either blue or white

dummy smart-phone) in MES, LH6 will receive information about what cover has been ordered. This is shown in Figure 5.2a.

Storage fetching: Here LH6 will drive to the storage facility and fetch a bottom cover (Figure 5.2b). A need for object localisation is needed, so LH6 is able to pick up the cover.

Bottom cover placement: LH6 will drive back to the dedicated FESTO module where it will perform a position calibration, to compensate for its $\pm$ 10 cm accuracy. It will then ask the PLC if a carrier has arrived, shown in Figure 5.3a. When the carrier has arrived LH6 will place the bottom cover on it (Figure 5.3b) and send a signal to the PLC that the bottom cover is placed.

Top cover placement: After sending the signal to the PLC, LH6 will drive to the storage facility and pick up a top cover, drive back to FESTO CP Factory and place it on a carrier (Figure 5.3c). These actions also use the position calibration and communication with the PLC used for the bottom cover.



(a) A signal is send to LH6 that it should go and fetch the blue parts.

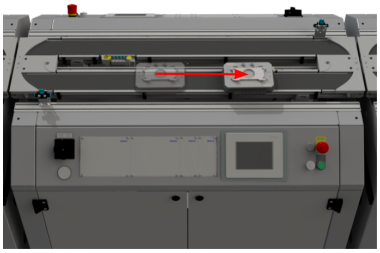
(b) An object localisation is carried out to identify an object.

Figure 5.2

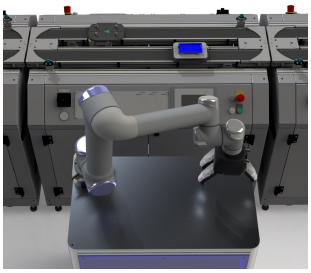
For this proposal to be implemented at the FESTO CP Factory, some hardware and software are required, which can be divided up into five subsystems as shown in Figure 5.4. Each of these subsystems contains different steps, which strives to solve the problem stated in Chapter 4.

Communication with FESTO CP Factory

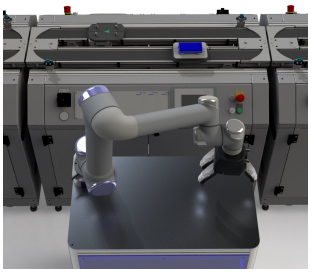
When ordering a dummy smart-phone, the first thing for MES to do is to



(a) The empty carrier arrives at the stopper and its RFID tag is read.



(b) LH6 drives back to FESTO CP Factory and place the bottom cover.



(c) After FESTO CP Factory is done working with the cover, LH6 puts on the top cover.

Figure 5.3

assign the order to a free carrier. However, since LH6 is going to place the cover, and not a dispenser module, communication between LH6 and FESTO CP Factory is needed. Chapter 6 describes how this is achieved.

Camera Calibration

When LH6 receives an order it has to drive to the storage facility in order to pick up a cover. To use the camera for detecting phone covers, the camera has to be calibrated in order to determine the intrinsic- and extrinsic parameters as well as removing distortion, which is described in Chapter 7.

Pose Estimation

In order to relate the image coordinates of an object to real world coordinates, the intrinsic parameters of the camera are used to create a projection from the camera to a given plane. This is described in Chapter 8.

Detection of Phone Covers

To detect the right cover, means that an image processing program has to be developed. This includes the traditional image processing steps described in Chapter 3. Chapter 9 gives an in-depth description of how this is done.

Calibration to FESTO CP Factory

The LH6's mobile position system makes it unreliable upon arrival at FESTO CP Factory. For the system to be able to adjust for the impressions of LH6, the camera can be used to calibrate the LH6 to a QR-code mounted on the module. Chapter 10 describes how this is achieved.

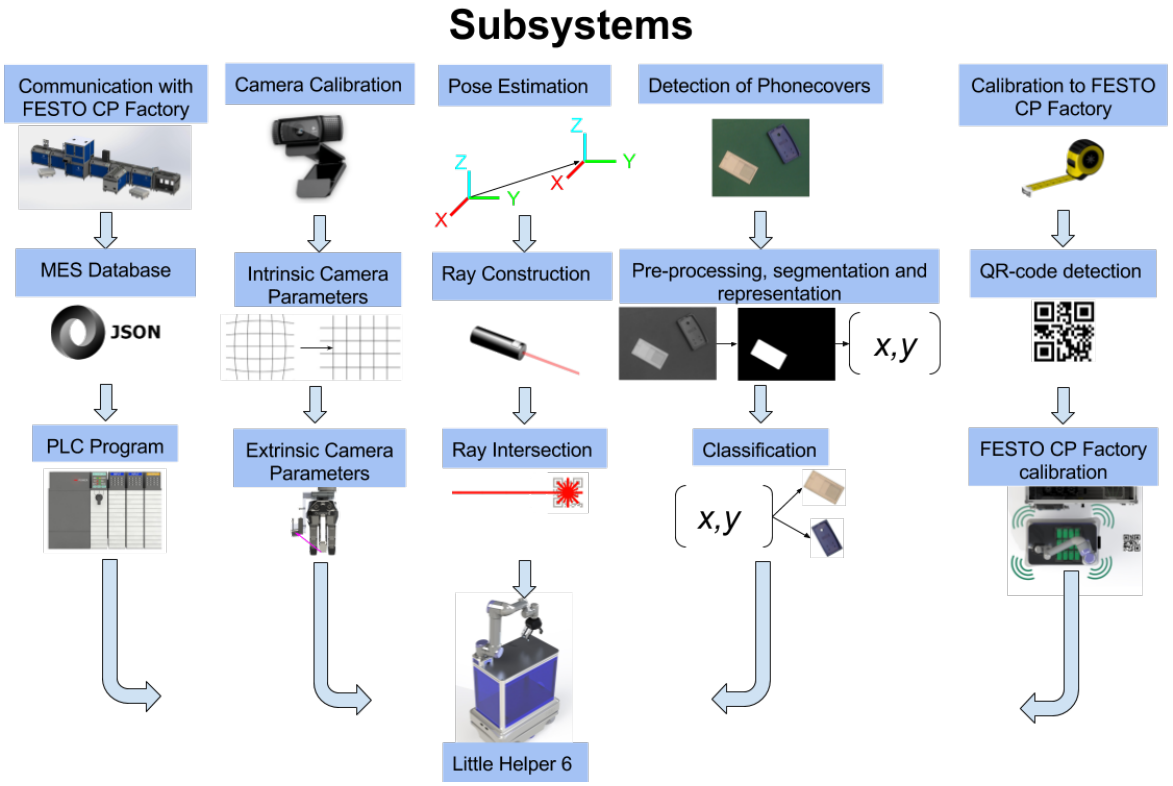


Figure 5.4: The five subsystems developed in Chapter 6 through 10.

Chapter 6

Communication with FESTO CP Factory

In Section 2.1.2 it was discussed the way communication flows within MES of FESTO CP Factory. This section describes how communication is established between the LH6 and FESTO CP Factory.

6.1 Communication with MES Database

To reach the requirement of a production cycle of 560 sec. MES has to communicate with LH6 immediately after the order is placed. Since none of the PLC's store any information related to orders, the communication should come from MES directly. However, due to the limited capabilities of MES, an external program is developed to send the MES database information to LH6.

The database of MES is a Microsoft Access database (accdb). In this database there exists over 60 tables, each containing information of FESTO CP Factory e.g. modules and orders. In table "tblOrderPos" all new production orders are placed. Here information about time and date, along with a unique ID and part number is stored. A part number is assigned for a special production order, which in this project is part number 6003 and 6004 for a blue and white dummy smart-phone respectively.

A C# program was developed with the function of reading *tblOrderPos* and publishing it to a web-server in a JSON format. In Listing 6.1 is an example on some the entries returned by the web-server with one order in *tblOrderPos*.

```
"MesData": {
  "TableOrderPosition": [
    {
```

```
        "OrderNumber": 2481,
        "Start": "16-03-2017 12:55:11",
        "End": "16-03-2017 12:55:22",
        "WorkPlanNumber": 14,
        "ResourceID": 65,
        "OperationNumber": 200,
        "PartNumber": 6004,
        "Error": false
        ...
      }
    ]
}
```

Listing 6.1: Web-server JSON.

This web-server is available to all clients on the same LAN through port 18081. The use of the information in Listing 6.1 will be explained in Chapter 11.1
For LH6 a ROS service driver, named "websocket_mes" was created that can request data from the web-server and publish the information to the ROS environment. This enables LH6 to start collecting the needed part(s) even before the production order is assigned to a carrier.

6.2 PLC Program

The PLC on the dedicated module needs to know, if LH6 has finished the operation on the module. To accomplish this a TCP/IP communication was established between the PLC and LH6.
When a carrier arrives at a module, the PLC gets a confirmation or denial from MES if it should do its operation, as discussed in Section 2.1. This is used such that the PLC knows if it should execute a program called *AppModul*. For this project a custom AppModul program has been created to enable the communication between the PLC and LH6. In Figure 6.1 this program is illustrated.

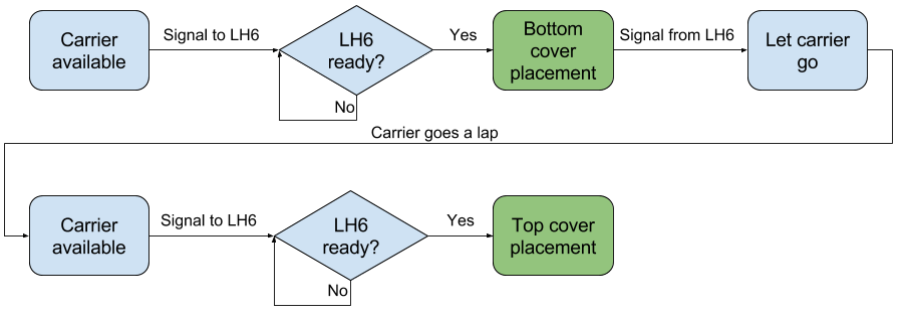


Figure 6.1: Flowchart of the AppModul program running on the PLC. The colour blue illustrates the PLC and green illustrates operation done by LH6.

Chapter 7

Camera Calibration

In order to construct a good hand-eye coordination between the object the manipulator has to pick up and the tool of the manipulator, it is necessary to make two calibrations in order to obtain the intrinsic and extrinsic parameters. This chapter gives an overview of what these two calibrations mean and how they are carried out.

Before any calibration is done a camera has to be selected. For this project an RGB Logitech HD C920 webcam is used (see Figure 7.1). The camera has different settings that can either be left on auto or manually be set. These settings makes it possible to adjust the camera to best capture the images for specifically this project. For a list of the camera settings, see Table 7.1, which also shows the values that the camera was set to. If the values of Table 7.1 are not chosen to be constant, the

Figure 7.1: Logitech HD C920 Webcam. (Logitech 2017)

image will change depending on what colours are in the image due to the camera trying to get a clear image. This is not desirable as it means the image processing will not be able to predict the colours in the image. An example is if the exposure is too large, the image will be over exposed and thereby removing features in the image that is desirable to keep. Figure 7.2 illustrates this principle. Another adjustment done to obtain good quality images, not directly related to the camera, is the scene in which the image is taken. A green fabric background is chosen in order to have a constant background, eliminating glares and maintaining a good

Setting	Default	Adjustment	Set value
Diagonal field of view	78°	-	-
Real optical resolution	$3MP$	-	-
Auto focus	enabled	disabled	-
Auto white balance	enabled	disabled	6000
Auto exposure	enabled	disabled	30

Table 7.1: Logitech HD C920 specifications. (Logitech 2017)

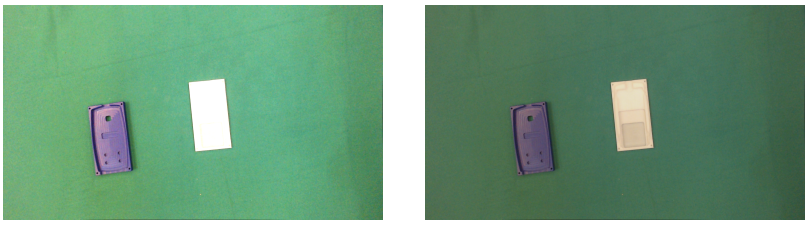


Figure 7.2: An over exposed image scene to the left and a correct exposed image to the right. Note, that details are kept in the correct exposed image.

contrast to blue and white covers.

7.1 Intrinsic Camera Parameters

The projection a camera produces from a 3D world coordinate to a 2D image point can be represented using the central perspective imaging model illustrated in Figure 7.3. A ray going from the world origin to the point $P = (X, Y, Z)$ will cross the image plane at the point $p = (u, v)$ and $z = f$ where f is the focal length. This can be shown using similar triangles:

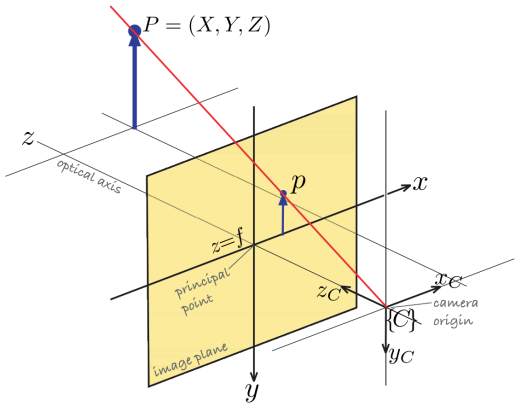


Figure 7.3: Illustration of the projection from 3D to 2D. (Corke 2011)

$$\begin{bmatrix} \frac{u}{f} \\ \frac{v}{f} \end{bmatrix} = \begin{bmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \end{bmatrix} \tag{7.1}$$

Solving for u and v:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{bmatrix} \tag{7.2}$$

In homogeneous coordinates this becomes:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{7.3}$$

This assumes that the center of the image is at the principal point (the point where the principal line is perpendicular to the image plane). However, due to imperfections in the camera this is generally not true. So therefore an offset is added to Equation 7.2 and is denoted x_0 and y_0:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f\frac{X}{Z} + x_0 \\ f\frac{Y}{Z} + y_0 \end{bmatrix} \tag{7.4}$$

Which means that Equation 7.2 becomes:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{7.5}$$

Equation 7.5 is represented in real world distance units (i.e. meters). This means that the output $[u,\ v]^T$ is also represented in meters. Since this is a projection to an image which is represented by pixels, the physical size of the photosensors in the camera, called photosites, are used to scale $[u,\ v]^T$. The size of the photosites are denoted ρ_x and ρ_y and are also measured in meters. If the photosites are square, this scaling factor will be the same in both the x and y direction. However, in the general case they can be assumed to be rectangular. This scaling in the x and y direction is denoted $k_x = \frac{1}{\rho_x}$ and $k_y = \frac{1}{\rho_y}$ respectively:

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & C_x \\ 0 & \alpha_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad \text{Where} \begin{cases} \alpha_x = k_x f \\ \alpha_y = k_y f \\ C_x = k_x x_0 \\ C_y = k_y y_0 \end{cases} \tag{7.6}$$

Equation 7.6 is thus the intrinsic camera calibration matrix. (Corke 2011) (Hartley and Zisserman 2003)

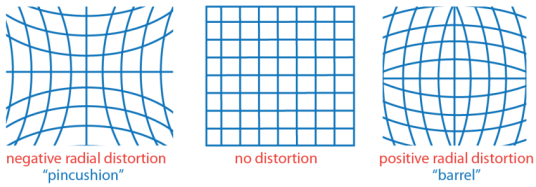


Figure 7.4: Example of distortion. (MathWorks 2014)

For this project an external program is used to determine this matrix. The program is found on the ROS website and is called camera calibration (`http://wiki.ros.org/camera_calibration`) and obtained the matrix (rounded to three decimals):

$$P = \begin{bmatrix} 1455.514 & 0 & 926.904 \\ 0 & 1449.646 & 582.076 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.7}$$

Another output of the ROS camera driver are the distortion coefficients. Ideally the lens of a camera does not alter the image, in reality this is hard to avoid. This alteration is called geometric distortion and contains two types of distortion: radial and tangential. Radial distortion affects straight lines so that they may not be straight near the edges of the image, but instead curve inwards or outwards. An example of radial distortion is shown in Figure 7.4 where lines curving inwards are called negative radial distortion, or pincushion distortion, and lines bending outwards are called positive radial distortion, or barrel distortion. Tangential distortion happens when the lens and the image plane are not perfectly aligned. The distortion coefficients were found to be:

$$\text{Distortion coefficients} = [0.0608, -0.165, 0.002, 0.002, 0] \tag{7.8}$$

7.2 Extrinsic Camera Parameters

The extrinsic camera calibrations is the transformation between the manipulator and the camera origin. The transformation is necessary in order for the manipulator to relate all its movements to the camera coordinate system and ultimately be able to pick up phone covers.
In order to calculate the transformation from tool to camera, it is necessary to have a world reference frame to which it is possible to get the transformation from the camera. This can be achieved by having a checkerboard with known size and three circles, as in Figure 7.5, and calculating the transformation from the camera to the three circles. Once it is possible to get the transformation from the camera to the world, and a transformation from the manipulator base to the manipulator tool,

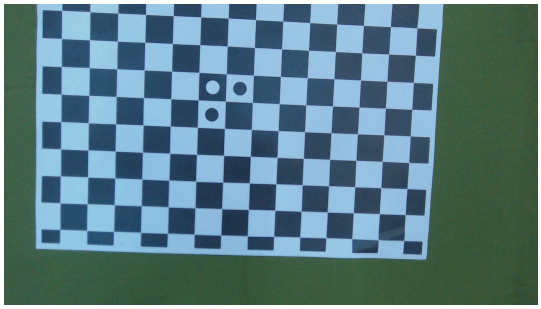


Figure 7.5: Extrinsic calibration board.

it is necessary to move the robot at least twice in order to get two different transformations both for the base to tool and for the camera to world. This gives the equation:

$$ {}^{t_1}_{t_2}T X = X {}^{C_1}_{C_2}T \tag{7.9} $$

$$ {}^{t_2}_{t_3}T X = X {}^{C_2}_{C_3}T \tag{7.10} $$

Where ${}^{t_1}_{t_2}T$ and ${}^{t_2}_{t_3}T$ are the transformations between two measured tool-frames and ${}^{C_1}_{C_2}T$ and ${}^{C_2}_{C_3}T$ are the transformations between the two measured camera-frames. X is the transformation between the tool and the camera and is thus the unknown to be solved for as shown in (Shiu and Ahmad 1989).

For this project the external programs DLR CalDe and DLR CalLab are used. DLR CalDe marks the squares from Figure 7.5 as landmarks and saves them to a file which DLR CalLab will then use to estimate the intrinsic parameters as well as the distortion. For each image there is a corresponding transformation from the manipulator base to the tool, which DLR CalLab uses with the estimated transformation from camera to world to estimate the tool to camera transformation. The calibration plate used as world reference for the camera is shown in Figure 7.5.(Strobl and Hirzinger 2006)

Chapter 8

Pose Estimation

One way to calculate the position of a point in a 3D world coordinate system based on a 2D image point, is to use the same similar triangles as mentioned in Chapter 7. That means the principal axis has to be perpendicular not only to the image plane but also to the plane on which the 3D point lies, before Equation 7.1 can be applied. In general this is very hard to ensure, and therefore it is not the method used in this project.
Instead, the method used in this project is to calculate a ray, based on the camera origin and a given image point on the image plane. This ray can then be used to calculate the intersection with a plane, i.e. the table, which will give the real world coordinates. This solution solves the problem of being dependent on the camera being exactly perpendicular to the table, as long as the table plane can be described in the camera coordinate system.

The coefficients of the ray is calculated using the camera matrix from Chapter 7 describing the intrinsic parameters of the camera. (Hartley and Zisserman 2003)
This matrix is gained by calibrating the camera and has the form:

$$P = \begin{bmatrix} \alpha_x & 0 & C_x \\ 0 & \alpha_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \tag{8.1}$$

Where α_x and α_y are the focal length and $[C_x\ C_y]^T$ is the principal point, i.e. the point where the camera point is perpendicular to the image plane.
This matrix can be used to map real world coordinates to the image plane by simple matrix-multiplications.

Since it is the opposite, i.e. the map from the image plane to the world, that is desired in this case, the pseudo inverse of P, denoted $P^\dagger$ ($P^\dagger = P^T(PP^T)^{-1}$ where

$PP^{\dagger} = I$), is used along with the position, x, on the image plane:

$$\overrightarrow{r}(t) = P^{\dagger}x \cdot t \tag{8.2}$$

Where $\overrightarrow{r}$ is the ray coordinates and t is the variable for the ray. This ray can then be used to intersect the real-world plane to get the original position of x before it was mapped to the image plane. This, however means that it is necessary to have the real-world plane described from the camera coordinate system. Figure 8.1 gives an overview of how the table plane can be related to the camera coordinate system. In order to describe the table plane, it is necessary to have the normal

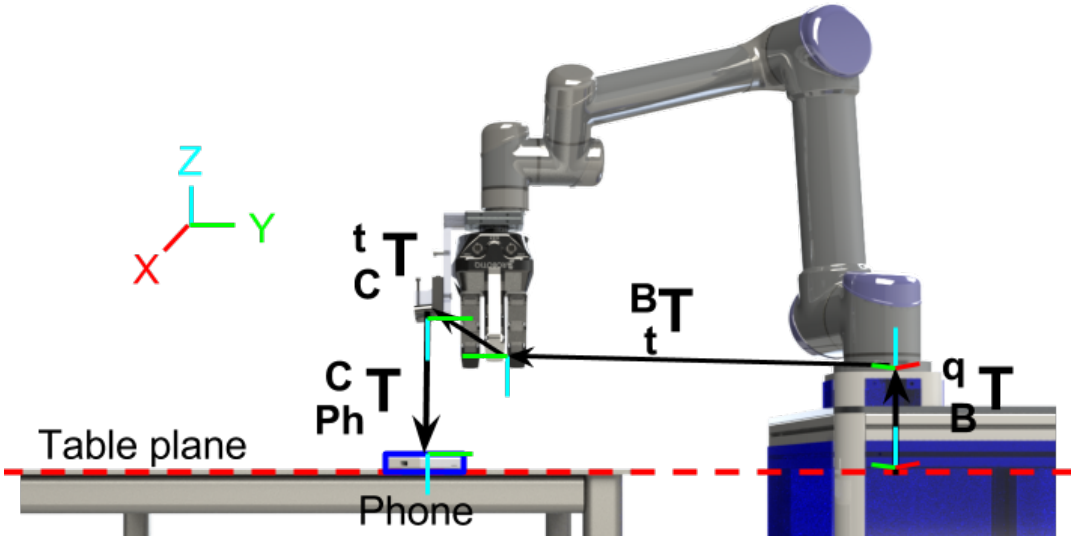


Figure 8.1: Transformation overview from the robot to the phone.

vector, $\overrightarrow{n}$, of the plane and a point on the plane, q. In a case where the table plane and the base of the robot is not parallel in the xy-plane, 3 points on the table are necessary but since the table plane is parallel to the base of the manipulator in the xy-plane the normal vector is given as $\overrightarrow{n} = [0,0,1]^T$. The point q can either be measured by hand or it can be measured by placing the manipulator tool on the table and taking the inverse of the transformation ${}^{B}_{t}T$, known from the kinematic model of the manipulator, and reading the translation:

$$q = \left({}^{B}_{t}T\right)^{-1} \cdot \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^{T} \tag{8.3}$$

This also means that the rotation matrix of the transformation from the point q to the base of the robot is the identity, ${}^{q}_{B}R = I_3$ and thus ${}^{q}_{B}T$ is:

$${}^{q}_{B}T = \begin{bmatrix} I_3 & q \\ 0 & \end{bmatrix} \tag{8.4}$$

After the manipulator has been moved to the camera position, q can be calculated as seen from the camera by taking the inverse of ${}^{q}_{B}T$, ${}^{B}_{t}T$ and ${}^{t}_{C}T$ multiplied and inversed:

$$ {}^{C}_{q}T = \left({}^{q}_{B}T {}^{B}_{t}T {}^{t}_{C}T \right)^{-1} \tag{8.5} $$

The rotation of ${}^{C}_{q}T$ can be used to calculate the normal vector of the table plane as seen from the camera.

$$ \overrightarrow{{}^{C}n} = {}^{C}_{q}R \, \overrightarrow{n} \tag{8.6} $$

q as seen from the camera is the translation of ${}^{C}_{q}T$:

$$ {}^{C}q = {}^{C}_{q}T \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^{T} \tag{8.7} $$

When the normal vector, $\overrightarrow{{}^{C}n}$, and a point, ${}^{C}q$ is known, it is possible to set up the equation for the plane using the equality $\overrightarrow{{}^{C}n} \bullet \overrightarrow{q} = \overrightarrow{{}^{C}n} \bullet \overrightarrow{u}$, where u is some random point on the plane $\overrightarrow{u} = \begin{bmatrix} u_x, u_y, u_z \end{bmatrix}^{T}$:

$$ {}^{C}n_1 {}^{C}q_1 + {}^{C}n_2 {}^{C}q_2 + {}^{C}n_3 {}^{C}q_3 = {}^{C}n_1 u_x + {}^{C}n_2 u_y + {}^{C}n_3 u_z \tag{8.8} $$

Note that the 4^{th} row of ${}^{C}q$ has been omitted. In order to get the intersection of the ray from equation 8.2 and the plane from equation 8.8, the ray is substituted into the plane equation instead of u:

$$ {}^{C}n_1 {}^{C}q_1 + {}^{C}n_2 {}^{C}q_2 + {}^{C}n_3 {}^{C}q_3 = {}^{C}n_1 r_1 \cdot t + {}^{C}n_2 r_2 \cdot t + {}^{C}n_3 r_3 \cdot t \tag{8.9} $$

Solving for t:

$$ t = \frac{{}^{C}n_1 {}^{C}q_1 + {}^{C}n_2 {}^{C}q_2 + {}^{C}n_3 {}^{C}q_3}{{}^{C}n_1 r_1 + {}^{C}n_2 r_2 + {}^{C}n_3 r_3} \tag{8.10} $$

When t is substituted into Equation 8.2, it will give a vector with the x, y and z coordinates on the table plane for the point on the image plane x mentioned earlier in this section. Figure 8.2 illustrates the principles of the image and table plane intersection. Note that the construction of the normal vector is omitted and is set to be $[0, 0, 1]^{T}$ (i.e. the table plane and the image plane are parallel in the xy-directions).

8.1 Rotation of Objects

In order to get the rotation of an object (i.e. phone cover or QR-code) another point is used as an orientation reference to calculate the rotation around the z-axis

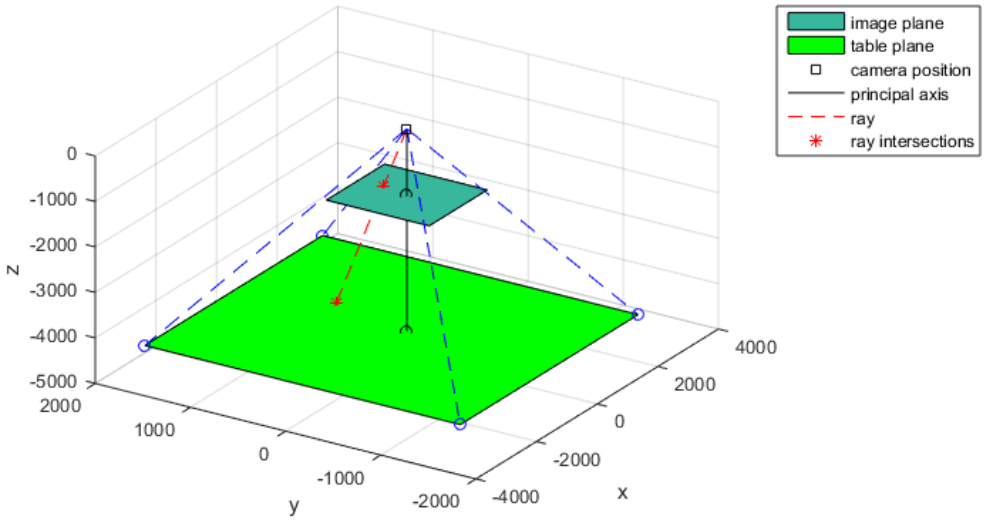


Figure 8.2: Illustration of intersection of a point on the image plane projected down on a table plane using a ray.

using the tan^{-1} function (see equation 8.11). The inputs for this function are the image coordinates, which means that it will fail if the camera is not approximately perpendicular to the table plane:

$$\theta_z = tan^{-1}\left(\frac{C_y - R_y}{C_x - R_x}\right) \tag{8.11}$$

Where C is the center of the object on the image plane and R is the orientation reference point. Only the rotation around the z-axis is considered in this project since all points of the image plane is assumed to lie on a plane approximately perpendicular to the camera.

The final transformation will thus look like:

$${}^{C}_{P}T = \begin{bmatrix} cos\,(\theta) & -sin\,(\theta) & 0 & r_x \\ sin\,(\theta) & cos\,(\theta) & 0 & r_y \\ 0 & 0 & 0 & r_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.12}$$

This transformation can then be used to get a transformation that can be sent to the manipulator (see Equation 8.13) so it can move to the point where the ray and the table intersects, as shown in Figure 8.1.

$${}^{B}_{Ph}T = {}^{B}_{t}T\,{}^{t}_{C}T\,{}^{C}_{P}T \tag{8.13}$$

This procedure is then tested on a chequerboard in the following section.

8.2 Subresults Pose Estimation

In this section the results of a test carried out to evaluate the precision of an implementation of the method from Chapter 8 is described. The test description is listed in the following:

1. Teach a point on the table plane using the manipulator tool in order to get ${}^{q}_{B}T$ as in Equation 8.4.
2. Place the camera, on the manipulator, above the table plane and facing towards the table.
3. Mark a pixel on the image of the chequerboard (see Figure 8.3).
4. After the manipulator has move to the given position, mark the position of the manipulator tool on the chequerboard.
5. Repeat step 2-4 25 times.

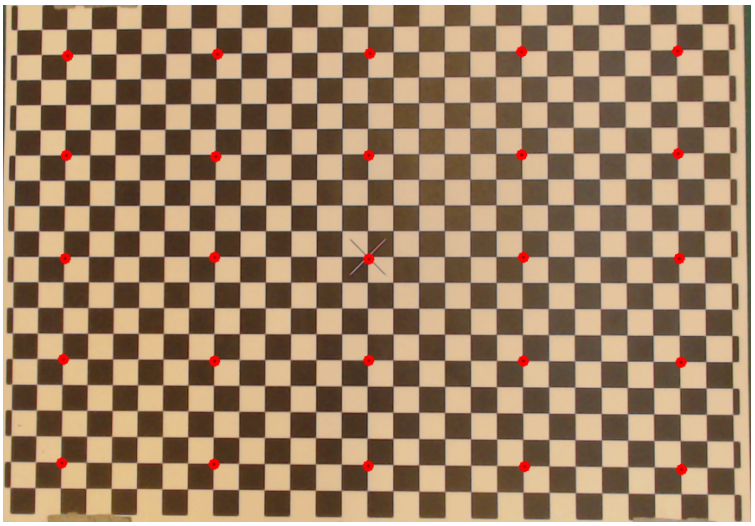


Figure 8.3: Targets indicated with red.

It can be seen from Figure 8.4, that the precision of the manipulator increases, the further to the right and down on the chequerboard the target is. Furthermore, all hits are placed in the lower right corner of the red circles except from the rightmost column. In general this indicates that the distance between the target and the hit is at the lowest when the manipulator is operating below row two and between column 20 and 26. Table 8.1 lists the mean and standard deviation of the measured error and shows that the LH6 may come into trouble if the target is located in the upper left part of the image. This increased error can be due to one or multiple

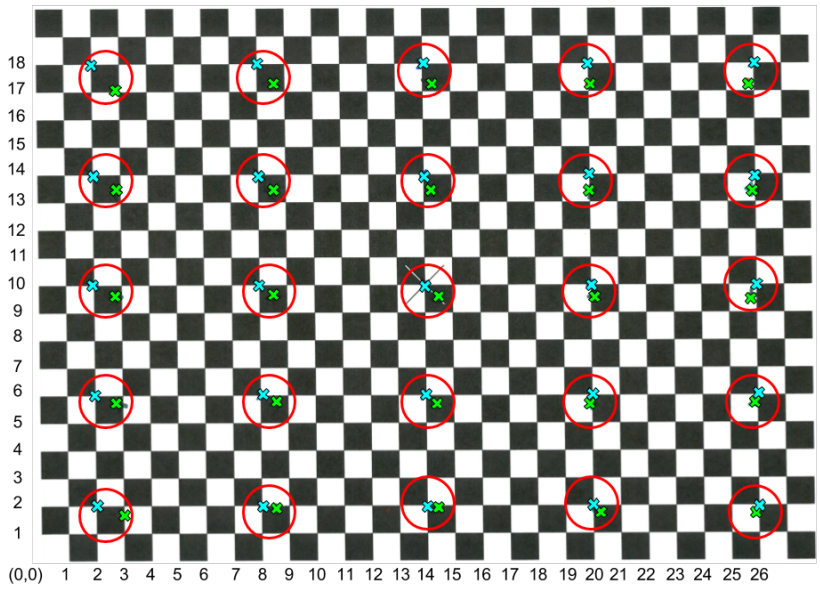


Figure 8.4: Results of the test, blue = target and green = actual hit. The red circles corresponds to a test pair (one target and one hit). Each square is 20x20 mm.

	Measure error [mm]
Mean	8.22
Std. Dev.	3.48
Max. Error	16

Table 8.1: Mean and standard deviation of the measured error.

factors such as imprecise camera calibration, hand-eye calibration and the table not being perfectly parallel with the base of the manipulator. Thus the implementation of the method does not fulfil the requirement of a position estimation of ± 5 mm, as the precision was evaluated to be ± 16 mm (see Figure 8.1).

Chapter 9

Detection of Phone Covers

This chapter consists of a description of the algorithm used to detect phone covers. The algorithm take point in the traditional image processing steps presented in Chapter 3 (Section 9.1-9.4) with the addition of finding the orientation of the phone covers (Section 9.6) and checking their accessibility (Section 9.5). The aim is to present a robust algorithm with the potential of satisfying the object classification success rate on 99% (See Chapter 5) and to return the position of an ordered phone cover such that the method introduced in Chapter 8 is applicable.

9.1 Pre-processing

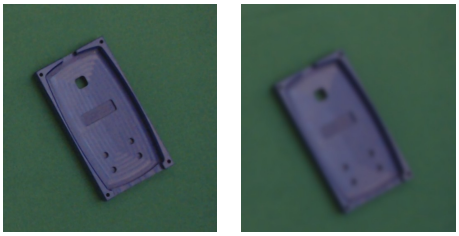
This section consists of an argumentation of the use of filters and considerations regarding colour space.

9.1.1 Gaussian Kernel

In order to reduce high frequency image noise, and at the same time enhance and save image structures, a filter using a Gaussian kernel will in this project be applied in the pre-processing phase. A Gaussian kernel is chosen because it weighs the center of its kernel more than its edges, and thereby maintains edge information of the image. The size of the kernel is obtained through an iterative trial and error process, and is identified to be 31x31 (see Figure 9.1).

9.1.2 Colour Space

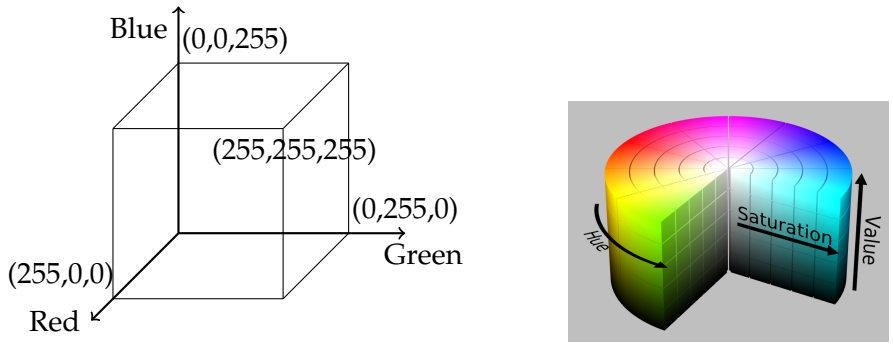
A colour pixel in an RGB image can be represented as a point within the RGB colour cube illustrated in Figure 9.2a. Considering this cube and the two pixels values (70,0,0) and (90,0,0), it is clear that they, are in the range (0,0,0) and (255,0,0). Meaning the two pixels both contains the same colour (red), but presented at different illuminations. The fact that the same colour space can be presented at different



(a) (b)

Figure 9.1: (a) An original image and (b) a blurred image using a Gaussian kernel on 31x31.

illumination and pixel values, can be a problem when identifying colours.



(a) (b)

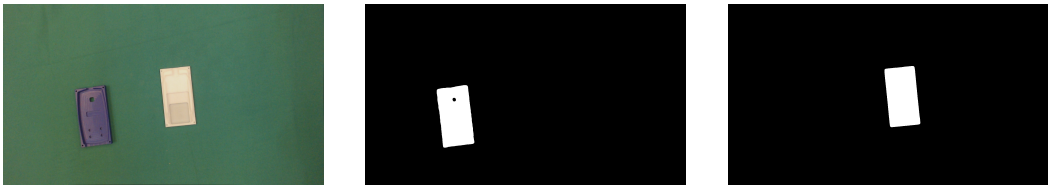
Figure 9.2: (a) The RGB colour cube and (b) the HSV colour cylinder. (Wikipedia 2017)

A solution to this problem, is to convert the RGB colour space to a space representing colours and illumination separately e.g. the HSV colour space (*Hue*, *Saturation*, *Value*). Using the HSV colour space it is possible to separate the pure colour (*hue*), represented as the angle in a circle, the purity of the colour (*saturation*), represented by a value between [0,1] and the illumination (*value*), represented as a value between [0,255] (see Figure 9.2b). It is then possible to use only *hue* and *saturation* to represent a colour, and thereby adding robustness to detection of coloured objects. However, when wanting to detect white objects, a combination of *saturation* and *value* can be used instead. Based on these considerations the HSV colour-space is assumed to be a robust and sufficient colour space for the application of detecting phone covers.

9.2 Segmentation

In order to segment the phone covers, from the background table, two HSV colour thresholds are applied separately on an input HSV image; a threshold for segment-

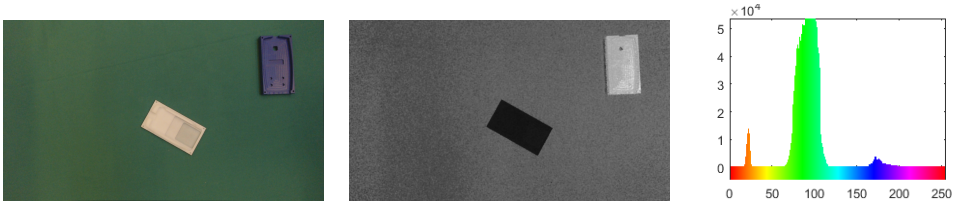
ing blue phone covers and one for white phone covers. The values within these threshold are denoted as potential covers, and represented by white pixels in a corresponding binary image. The result of this segmentation are two binary images; one image containing BLOBs of potential blue covers and one containing BLOBs of potential white covers (see Figure 9.3).



(a) **(b)** **(c)**

Figure 9.3: (a) An original image and its corresponding two binary image, where (b) is the blue binary image and (c) is the white binary image.

The histogram related to the *hue* channel of a blue cover image, illustrated in Figure 9.4, contains two peaks; a large peak spanning from 60-125 and a smaller spanning from approximate 160-200. In unscaled *hue* this corresponds to 84°-176° which indicate a green area, and 225°-282° indicating a blue area. Based on the histogram in Figure 9.4, a lower unscaled threshold on 190° and upper threshold on 300° is identified, for blue covers. A threshold for *hue* is for the blue cover assumed to be sufficient, thus the two other HSV channels are for the blue cover not defined.



(a) **(b)** **(c)**

Figure 9.4: (a) An RGB image of a blue cover, (b) the *hue* channel of the RGB image converted and (c) a pixel histogram of *hue* values related to a converted image. Note *hue* has been scaled from [0,360°] to [0,255].

A histogram of a *value* channel related to a white cover image, reveals two peaks; one big peak in the area, 75-125 and one smaller in the area 180-230 (see Figure 9.5). Here the values of the background are present in the span from 75-125, and the white cover in the span from 180-230. Based on these observations, a threshold for the *value* channel is chosen to span from 160-255. It is assumed that a threshold

	Hue		*Value*	
Type	Bottom Blue	Top Blue	Bottom White	Top White
Mean	136.4814	134.2381	219.5930	224.9671
STD	6.8477	5.6991	1.1529	0.6034
Threshold_{min}	237.6834	239.1142	216.6230	223.4128
Threshold_{max}	308.2423	297.8383	222.5629	226.5215

Table 9.1: Mean and Standard Deviation from 39 covers of each type.

for the *value* channel is sufficient for segmenting white objects, thus a threshold for the other HSV channels are not defined.

In order to verify the identified threshold of 190°-300° for the blue covers and 160-255 for the white covers, the average *hue* and *value* of 39 covers of each type (156 covers in total) are used to calculate a 99% confidence interval as shown in Table 9.1. This is assuming the *hue* and *value* are normal distributed. It can be seen from the table that the identified thresholds have a larger range than the confidence interval. This is due to the confidence interval being the average of the *hue* and *value* channel of 156 covers, where the identified thresholds should ideally capture all the pixels in an object.

Often morphology can be used after the segmentation to remove noise in the binary image. However, since there is a large gap between the blue and green in *hue* and white and green in *value*, this is assumed unnecessary.

(a) **(b)** **(c)**

Figure 9.5: (a) An RGB image of a white cover, (b) the value channel of the RGB image converted and (c) a pixel histogram of the *value* related to a converted image.

9.3 Representation of Phone covers

When the image is segmented the BLOBs in the image needs to be represented, in order to be classified. To represent the BLOBs several features are extracted from

each BLOB. The extracted features contains different information which can be used for different classifying tasks. For this project there are two major classifying tasks:

1. To classify if a BLOB is a phone cover or noise.
2. To classify if a phone cover is a top cover or bottom cover.

Features related to the geometry of the BLOBs are used to determine if the BLOB is noise or a potential phone cover. These features can be used while the distance from the table to the camera is constant (As specified in Chapter 5). The following description lists the features used to the first classifying task:

The radius of a minimum enclosing circle

The radius of a minimum enclosing circle of a BLOB, gives an indication the size of the BLOB. In OpenCV this can be implemented using the functions; **findContours** and **minEnclosingCircle**.

(a) **(b)** **(c)** **(d)**

Figure 9.6: (a) The original RGB image, (b) the corresponding binary image of a white cover, (c) drawn contours of the BLOB and (d) the minimum enclosing center, center of the BLOB and radius.

The area of a contour

Looking at the area of a contour is useful for filtering out objects which are too small or too big. The area can be calculated by looping through the contour points and summing the difference in either the x or y direction. For this project the contour area is found by using the OpenCV function; **findContourArea**.

The ratio between the longest and shortest side of a minimum area rectangle

By using the ratio between the longest and shortest side of a rectangle instead of using the length of the sides directly, the robustness of the feature can be increased since the ratio is less likely to change with small variations in the camera position. In order to find the minimum area rectangle of a BLOB, the BLOB has to be encapsulated in a convex-hull. Then placing a rectangle with sides parallel to the x and y-axis (up-right rectangle) around the convex BLOB and rotating it until one side coincides with an edge of the convex BLOB. The area is then calculated and the procedure of rotating the rectangle is continued until all edges of the convex BLOB has coincided once with the

rectangle. The rectangle with the smallest area is then found by selecting the rectangle that produced the smallest area. (Freeman and Shapira 1975)
In OpenCV a minimum area rectangle can be obtained using the function **minimumAreaRect** on a contour related to a BLOB (see Figure 9.7).

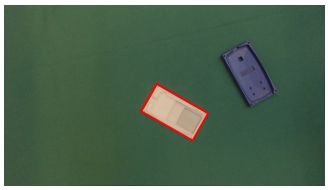


Figure 9.7: A minimum area rectangle obtained using a contour of a BLOB related to the white cover. The rectangle is found using the OpenCV function **minAreaRect**.

For the next classifying task two features are extracted, which separates bottom and top covers:

The area of an inner contour:

Looking at the BLOB of a top and bottom cover, it is clear that the most distinct difference is the hole in the bottom cover (see Figure 9.8). Based on this observation, it is assumed that the area of that hole is a good representation of it being a bottom cover. When finding the contour of a whole cover it is possible to use the information returned by the function **findContours**, to check if a BLOB has a hole or not. If a BLOB has a hole the function will return two contours (an outer and an inner contour), the area of the inner contour, is obtained using the function **contourArea**. If a hole is not present, **findContours** will return one contour and the inner contour area is set to 0. However, this feature has the downside of including noise if the BLOBs are not fully segmented.

(a) **(b)**

Figure 9.8: (a) An original image, (b) the corresponding binary image.

Amount of green pixels on the object of interest:

As illustrated in Figure 9.9, the hole in the bottom cover object contains green pixels. These green pixels are not present within the bottom cover. This

means that by counting the amount of green pixels within a cover it should be possible to extract a feature which makes it possible to separate top and bottom covers. In order to identify green pixels a threshold has to be set. This threshold is found using the *hue* channel and the same approach as in Section 9.2.

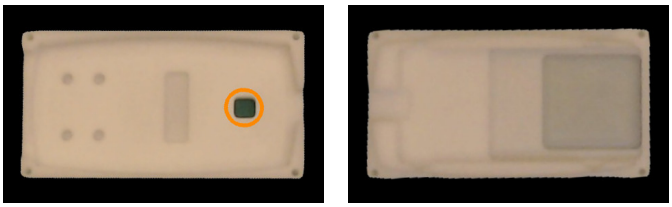


Figure 9.9: The region of interest for respectively top cover and bottom cover.

9.4 Classifications

When an image is segmented and the image BLOBs are represented by features, the BLOBs needs to be classified to determine, to which of the following classes they belong:

- Class 0: Blue bottom phone cover.
- Class 1: Blue top phone cover.
- Class 2: White top phone cover.
- Class 3: White bottom phone cover.
- Class 4: Not a phone cover.

This section consists of a description of how to solve the two classification tasks, described in Section 9.3. This is done by a presentation of one method for classifying if a BLOB is a cover or noise, and two different methods for classifying the type of cover a BLOB belongs to, given it is accepted as a cover.

9.4.1 Classifying Task 1: Accepting Covers

As a solution to the first classification task, where BLOBs are being classified as either noise (class 4) or being a phone cover, thresholds are identified, for the three features extracted in Section 9.3. If a given BLOB representation is within all these thresholds, it will be accepted as a cover. The features and their identified thresholds are listed in the following:

1. The radius of the minimum enclosing bounding circle related to a contour of a BLOB should, measured in pixels, be bigger than 170 and smaller than 210.
2. The contour area of the BLOB should, measured pixels, be bigger than 55000 and smaller than 70000.
3. The relation between the longest and the smallest side of the minimum rectangle which can be drawn around a given BLOB, should be bigger than 1.7 and smaller than 2.1.

9.4.2 Classifying Task 2: Type of Cover

As mentioned earlier two methods will in this section be presented for classifying the type (top or bottom) of a phone cover. As illustrated in Figure 9.10, and described in Section 9.2, the images will after the segmentation be divided into two binary images, one containing potential blue covers and another containing potential white covers. Furthermore the classification done in Section 9.4.1 has removed all noise. Thus the classifiers task in this section is to determine if a BLOB represents a top or bottom cover, and not if it is a blue or white cover.

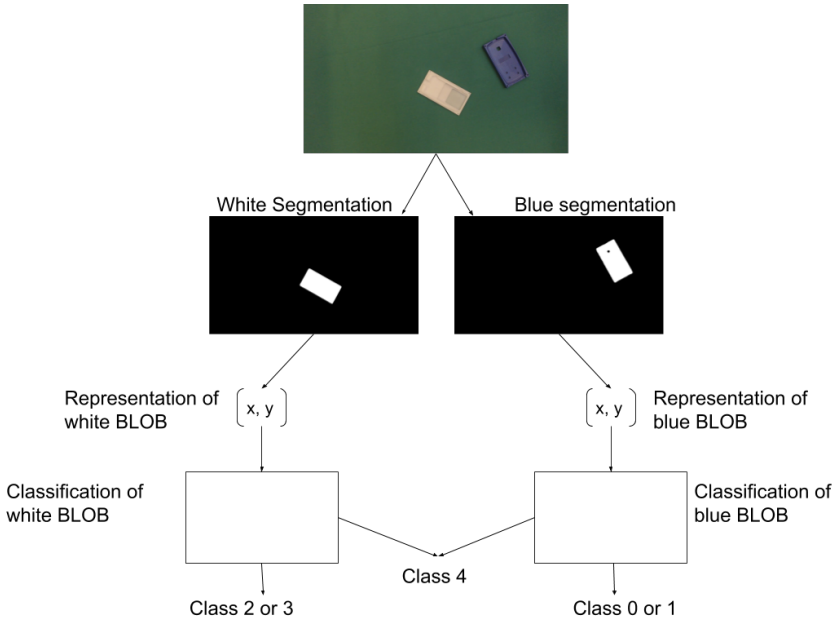


Figure 9.10: The different image processing steps leading down to the classification.

The first method presented consists of a classifier based on a simple threshold (TH). It is assumed that by associating BLOBs with an inner contour area above 10

to a top cover and below 10 to bottom cover, it is possible to classify the covers. While the method is only depending one feature the method will fail if a given cover does not meet the demands of the threshold.

The second presented method is a K-Nearest Neighbour (KNN) classifier, using the two features described in Section 9.3; inner contour area and the amount of green pixels within a cover. A KNN classifier is more complicated than a simple threshold, but given good training data, it is assumed too be more robust. A KNN

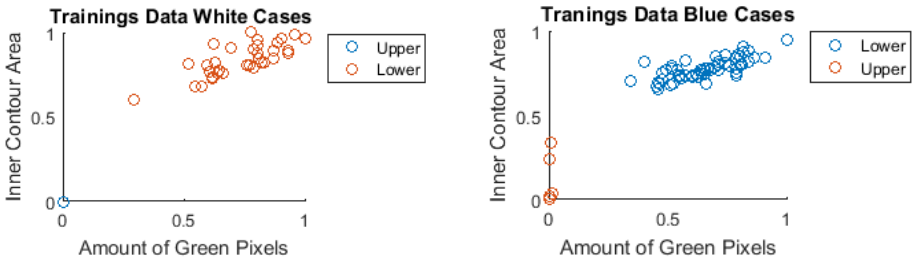


Figure 9.11: A scatter plot of the white trainings data and a scatter plot of the blue trainings data. Note, that the data are normalised.

classifies incoming features using the euclidean distance to a set of training data. To obtain training data and create a training model for the classifier, 50 images of each phone cover was captured. For each image, features were extracted, noted in a file and their associated class was labelled. A scatter plot of these data are displayed in Figure 9.11, where a good separations between the two classes is clear. Note that the training data for the white and blue covers are noted separately. When the training data is obtained, the data can be used to train the classifier and thereby be used as a model for new incoming features, following the concept of Figure 9.12.

In Section 9.7 the TH and KNN are compared in order to select the optimal one for this project.

9.5 Accessibility Check

After all the phone covers in a captured image are classified, the next step of the algorithm is to determine what cover to pick up. The accessibility check determines if any phone covers of a given class is free, by rotating and translating a generic gripper template with respect to the phone cover it has to check, as seen in Figure 9.13c. When the rotation and translating is done a logic AND operation is applied between a full binary image (see Figure9.13b) and the gripper template (see Figure 9.13c).

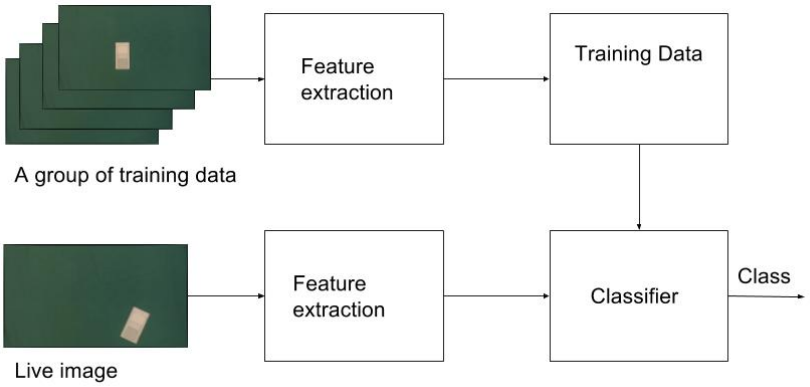


Figure 9.12: The concept of using training data as a model for new incoming data.

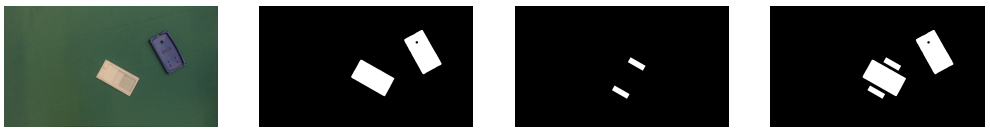


Figure 9.13: (a) An original RGB image, (b) the corresponding full binary image, (c) a gripper template and a full binary image and (d) the gripper template around the white top cover.

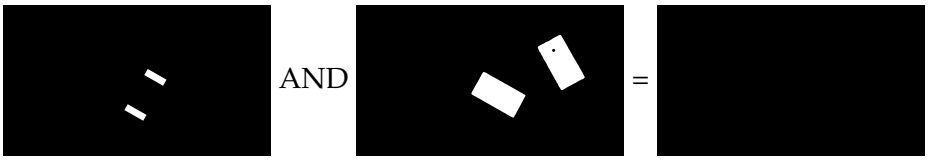


Figure 9.14: The result of an AND operation between a full binary image and a gripper template, when the given cover is accessible.

If the resulting image from the AND operation is a completely black image, the phone cover is accessible (see Figure 9.14), and if there are white areas in the resulting image the phone cover is not accessible (see Figure 9.15). The accessibility check will in a random order do this check for all phone covers of a wanted class, until a phone cover is determined to be accessible.

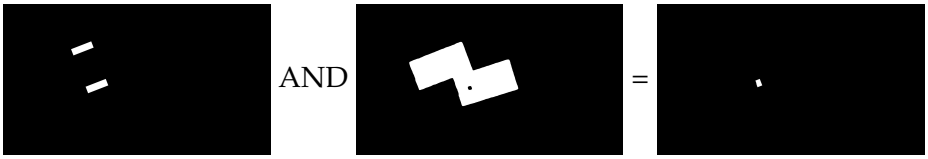


Figure 9.15: The result of an AND operation between a full binary image and a gripper template, when the given cover is not accessible.

9.6 Localisation of Covers

When picking op a cover, the pose estimation method introduced in Chapter 8 needs both a reference point and an orientation. The reference point is the center of a cover and the point the manipulator moves to (see Figure 9.16), and an orientation enabling the end effector to rotate such that it can grip around the narrow side of a cover.

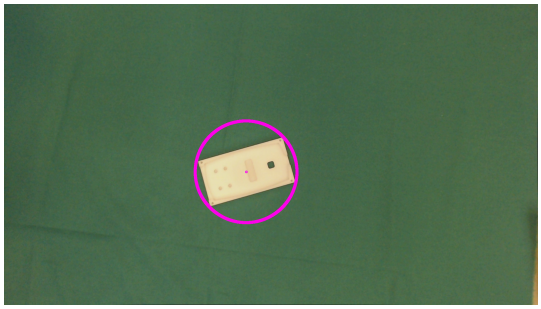


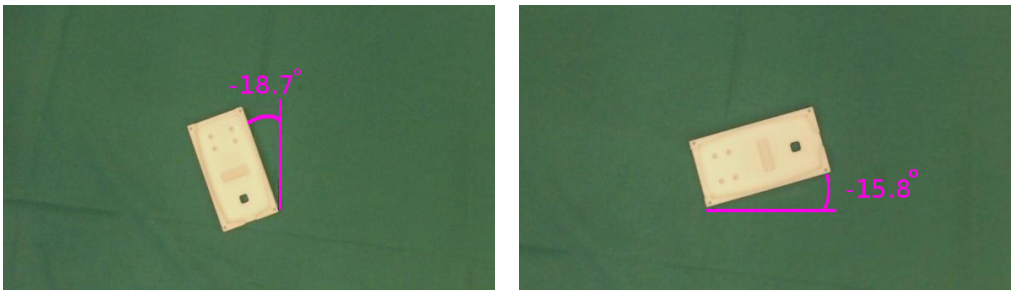
Figure 9.16: A boundary circle obtained using the OpenCV function **minEnclosingCircle** and a reference point (the center of the circle).

The reference point are in both bottom and top covers identified using the OpenCV function **minEnclosingCircle**, which returns the center of an enclosing bounding circle of a BLOB (see Figure 9.16). The orientation of a top and bottom cover are found by two different methods. The orientation of a top cover is required to be known (as stated in Section 5.1), whereas the only requirement of the orientation of a bottom cover, is that the gripper should be rotated such that it can grip around the narrow side of a bottom cover.

9.6.1 Orientation of Bottom Covers

When detecting the orientation of a bottom cover, the OpenCV function called **minAreaRect** is used. This function finds the minimum rectangle which can contain a BLOB, and returns a class called **RotatedRect** consisting of the attribute **angle** and **size**. The **angle** is the closest clock wise rotation a cover needs to rotate

in order to be an up-right rectangle (see Figure 9.17). The **size** contains information about the dimensions of the minimum rectangle. As illustrated in Figure 9.17, there are two scenarios for an upright rectangle. If the height is bigger than the width, the angle is accepted as it is and if the width is bigger than the height, an offset of 90° is added to the angle before passing it on.



(a) **(b)**

Figure 9.17: The two orientation different scenarios: (a) the height of boundary box is bigger than the width, (b) the width of the boundary box is bigger than the height.

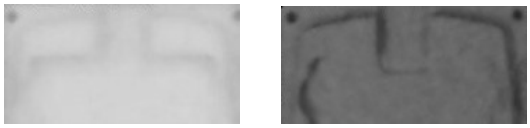
9.6.2 Orientation of Top Covers

By detecting and locating a point in a specific area of a cover and using this together with the reference point, the orientation can be calculated using simple geometric considerations, as shown in Section 8.1.

A method assumed to be capable of locating a specific area of a top cover, is template matching. By using an image template of the area and measuring the similarity between the template and a live captured image, the specific area can be identified.

The template used for this operation is the top area of a top cover. This area is the most unique area, having a distinct pattern (see Figure 9.18). As there are no significant colour differences between the upper and lower area, it is decided that the template should be obtained using the *value* channel of an HSV image. This is under the assumption that the value channel preserves edges better than hue and saturation.

For the template matching algorithm correlation is used, by using an image template as a kernel. Correlating an input image, $f(x,y)$, with a template, $t(x,y)$, results in an output image $g(x,y)$. The brighter values of $g(x,y)$ indicates a higher similarity between the image and the template. Mathematically, correlation is expressed as Equation 9.1, where R_x is the radius in x direction and R_y is the radius



(a) **(b)**

Figure 9.18: (a) The template for the white top covers, (b) the template for blue top covers.

in the y direction, of the template.

$$g(x,y) = \sum_{j=-R_y}^{R_y} \sum_{i=-R_x}^{R_x} t(i,j) \cdot f(x+i, y+j) \tag{9.1}$$

As illustrated in Equation 9.1, $g(x,y)$ is obtained by multiplying $t(x,y)$ and $f(x,y)$, meaning that possible bright areas in $f(x,y)$, might produce high values in $g(x,y)$. This poses a problem e.g. reflected light can affect the result of the template matching.

For this project a method for normalising the output and making it independent of the level of light in an image, is used to solve this problem. The normalisation is done using the correlation, the length of the template and the length of the image path. Resulting in a template matching method, which is invariant to changes in intensity, called Normalised Cross Correlation (NCC). The similarity score of a output image in NCC, is within the interval [-1,1], where 1 is a perfect match. NCC is given by Equation 9.2. (Corke 2011) (Moeslund 2012)

$$g(x,y) = \frac{\sum_{j=-R_y}^{R_y} \sum_{i=-R_x}^{R_x} t(i,j) \cdot f(x+i, y+j)}{\sqrt{\sum_{j=-R_y}^{R_y} \sum_{i=-R_x}^{R_x} t(i,j)t(i,j)} \cdot \sqrt{\sum_{j=-R_y}^{R_y} \sum_{i=-R_x}^{R_x} f(x+i, y+j)f(x+i, y+j)}} \tag{9.2}$$

To carry out the template matching algorithm, the OpenCV function, **matchTemplate** is used. An assumption implemented in this function is that the given covers, are approximately the same orientation as the template. This assumption is in most covers violated, since the orientation of the covers are unknown and random. Therefore, several steps are undertaken to secure that the orientation, of a top cover is the same, every time before applying template matching. The steps of the algorithm can be found in the following list and are illustrated in Figure 9.19:

1. An image is captured.

2. A top cover is extracted from the original image.
3. The cover is cropped and split in two half covers.
4. The half covers are rotated and converted to the *value* channel of HSV.
5. A template matching operation is applied separately on both of the half covers.
6. The output image containing the highest maximum similarity value is accepted as the best match.

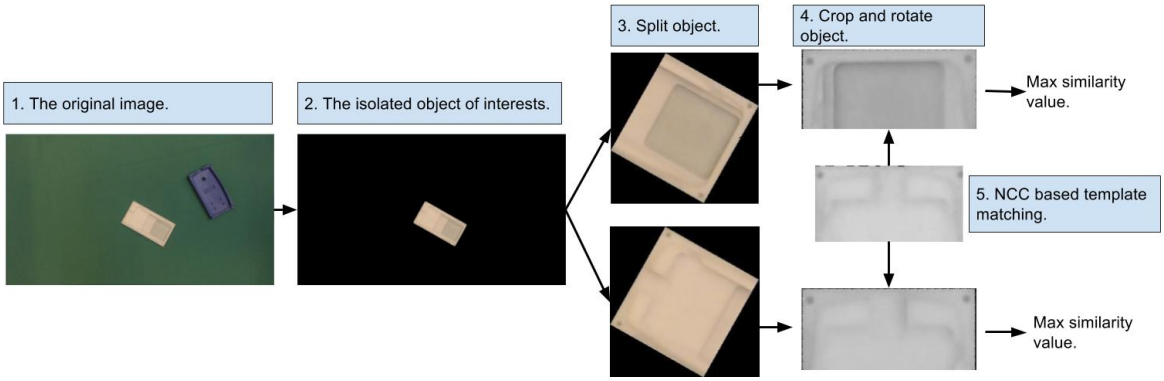


Figure 9.19: The steps undergoing when doing template matching.

Once the half cover with the best match is found the center of the half cover in the original image can be used to calculate the rotation point. Thereby it is possible to obtain two points related to the object (see Figure 9.20):

1. The reference point obtained in Section 9.6.
2. The center point related to the half cover of the best match.

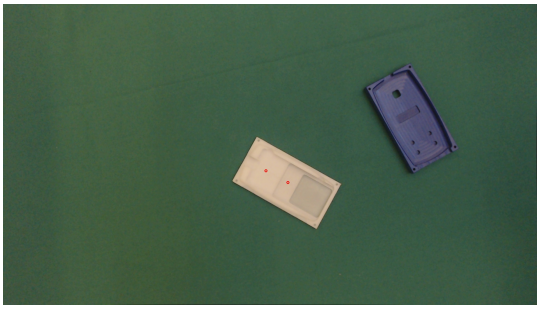


Figure 9.20: The reference point (the center of the whole cover) and center point of the half cover related to the best match.

9.7 Subresults for Object Detection

This Section consists of results obtained from testing the algorithm described in the Chapter 9. The test consists of two sub-tests and take point in the same setup (illustrated in Figure 9.21) and are described in the following:

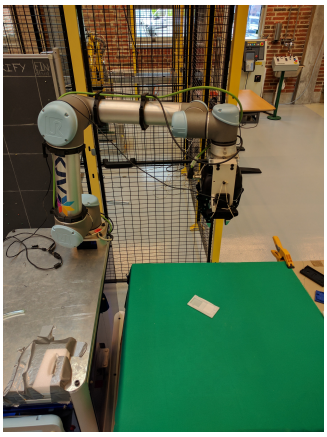


Figure 9.21: The test setup for the evaluation of the object detection algorithm.

Evaluation of Classifier for Phone Detection
This test consists of an evaluation of the two classification methods and the features applied (see Section 9.4.2). The classifier that evaluates best will be used as a classifier for the solution. The test consists of four test iterations spread evenly out over four hours. Each iteration consists of 50 classification tries for each class (see Section 9.7.1).

Evaluation of Top Cover Orientation
This test consists of an evaluation of the NCC based template matching algorithm used to detect the orientation of the top covers (see Section 9.6.2). The test consists of four test iterations, where the algorithm is applied on 25 live captured image of each colour (see Section 9.7.2).

9.7.1 Evaluation of Classifier for Phone Detection

The following test consists of a performance evaluation of the presented classifiers; KNN and a simple threshold using inner contour area (TH) (see Section 9.4). The performance of the classifiers will be evaluated by tests carried out in four iteration with a hours span in between. The first iteration started at noon and ended four hours later in the afternoon. For the KNN classifier the test was initialised by creating a set of training data consisting of data from 50 images of each cover class, and identifying a suitable thresholds for the feature counting green pixels. For the

TH method a threshold for the inner contour area was identified. These initialised values were kept static throughout the test.
The success rate for the classification is determined, in order to secure that the classifiers are able to fulfil the required success classification rate on at least 99% (See Section 5.1).

Before the test, the UR5 manipulator is moved to a static position above the table where the phone covers will be placed (see Figure 9.21). The test description for a test iteration is listed in the following:

1. Place a cover in a random position and orientation on the table, and secure it is within the cameras field of view.
2. Use the classification method, together with the algorithm described in Section 9, to classify a cover and note the result.
3. Step 1,2 are repeated 50 times for each cover class and classifier.

The overall classification success rate of the classifiers, showed that TH had a success rate on 99.6% and thereby performed better than the KNN, with an overall success rate on 98% (see Table 9.2 and Table 9.3). Based on this, it can be concluded that TH was the only classifier to reach the requirement of a classification rate on 99%.

The classification of the white covers conducted in general a higher classification

Iteration nr. and time	**Class 0**	**Class 1**	**Class 2**	**Class 3**	**KNN**
nr. 1 - 12:00	100%	98%	100%	100%	99.5%
nr. 2 - 13:00	90%	100%	100%	100%	97.5%
nr. 3 - 14:00	80%	100%	100%	100%	95%
nr. 4 - 15:00	100%	100%	100%	100%	100%
Overall success rate	92.5%	99.5%	100%	100%	98%

Table 9.2: The test results for the KMM classifier.

success rate than the blue covers, for both TH and KNN. The overall classifications of the blue top cover had 1 false negative classification as a blue bottom cover, for KNN and 3 for TH. Where the bottom blue cover conducted 15 false negative classifications as a blue top cover, for KNN and 0 for TH. There were furthermore no covers falsely classified as being noise (class 4) (see Figure 9.4).
In order for a top cover to be classified as a bottom cover using the TH method, the top cover must have an inner contour area greater than the set threshold. This can only occur due to noise in the image and as it can be seen in table to the right in Table 9.4, the blue top phone covers are more prone to this happening. A solution

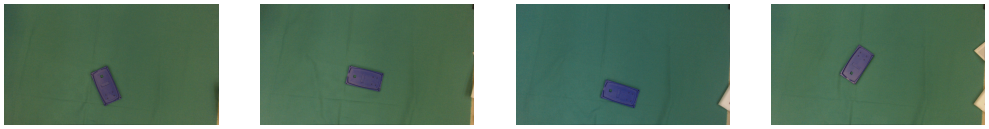
Iteration nr. and time	Class 0	Class 1	Class 2	Class 3	Threshold
nr. 1 - 12:00	100%	98%	100%	100%	99.5%
nr. 2 - 13:00	100%	98%	100%	100%	99.5%
nr. 3 - 14:00	100%	100%	100%	100%	100%
nr. 4 - 15:00	100%	98%	100%	100%	99.5%
Overall success rate	100%	98.5%	100%	100%	99.6%

Table 9.3: The test result using a threshold on the inner contour feature.

to this might be to consider increasing the threshold, or adding a morphology filter after the segmentation phase.

As it can be seen in the first column (class 0) of Table 9.2, the success rate drops over time with a minimum at the third iteration, after which it increases again. The same phenomenon is not present for the white bottom covers (class 3). This could indicate that the amount of green pixels on the bottom blue cover varies significantly with the amount of light in the scene. The change in light between each iteration can be seen in Figure 9.22c.

Due to these results the TH method is further used for classification in this project.



(a) **(b)** **(c)** **(d)**

Figure 9.22: The changes of light in the scene throughout the test iterations where, (a) first, (b) second, (c) third and (d) fourth iteration.

9.7.2 Evaluation of Top Cover Orientation

In order to secure that the orientation of the top covers are known, a test is applied using the template matching algorithm described in Section 9.6.2. In order to fulfil the required success rate for the top cover orientation, it is required that the template matching algorithm have to obtain a success rate on atleast 99% (see Chapter 5). The performance of the algorithm will be tested in four test iterations consisting of 25 tests for each colour, carried out with one hour in between, starting from the noon and ending in the afternoon. This was done in order to evaluate the performance throughout a day, as light changes. The test will focus on the template matching success rate, and the maximum similarity score in the output

0	185 23.1%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	99.5% 0.5%
1	15 1.9%	199 24.9%	0 0.0%	0 0.0%	0 0.0%	93.0% 7.0%
2	0 0.0%	0 0.0%	200 25.0%	0 0.0%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	0 0.0%	200 25.0%	0 0.0%	100% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
	92.5% 7.5%	99% 0.5%	100% 0.0%	100% 0.0%	NaN% NaN%	98.0% 2.0%
	Class 0	Class 1	Class 2	Class 3	Class 4	

0	200 25.0%	3 0.37%	0 0.0%	0 0.0%	0 0.0%	99.85% 0.15%
1	0 0.0%	197 24.63%	0 0.0%	0 0.0%	0 0.0%	100% 0.0
2	0 0.0%	0 0.0%	200 25.0%	0 0.0%	0 0.0%	100% 0.0%
3	0 0.0%	0 0.0%	0 0.0%	200 25.0%	0 0.0%	100% 0.0%
4	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
	100% 0.0%	98.5% 2.5%	100% 0.0%	100% 0.0%	NaN% NaN%	99.6% 0.4%
	Class 0	Class 1	Class 2	Class 3	Class 4	

Table 9.4: Two confusion matrices: (to the left) contains information obtained the KNN classifier and (to the right) contains information obtained using the TH classifier. In the tables the x-axis is the false negative and the y-axis is the false positive.

values. For each test there are two scores; one for the area which is achieved to be detected and one for other area. Before the test the UR5 manipulator is moved in the specified hight above the table where the phone covers will be placed (see Figure 9.21). The specification for the test are listed in the flowing:

1. Place a top cover of a certain class on the table, within the field of view of the camera and in a random position and orientation.

2. Use the template matching algorithm described in Section 9.6.2 to determine the orientation of the cover.

3. Note, the two max similarity scores, and the result of the match.

4. Repeat step 1, 2 and 3, 25 times for each top cover class.

The test showed that the template matching function managed to detect the orientation of white top covers, with a success rate of 100%. The blue top covers had 8 false detection and thereby managed to reach a success rate on 92% (see Table 9.5). This indicates that the template matching applied on the white covers, in general is more stable than the one applied on the blue covers. It was discovered that false matches of the blue covers happened more frequently in certain areas of the table, indicating the white template is more robust against e.g. shadows. As it can be seen in Table 9.6 the mean of the correct area and wrong area are very similar, indicating both areas are good matches to the template. However, due to the significantly small standard deviation, the matching scores do no overlap in most of the cases, explaining the overall classification success rate on 96% shown in Table 9.5.
Thus the algorithm does not fulfil the requirement of a 99% classification success rate. The assumed cause for this, is that the correct half cover and wrong half cover

Test iteration nr.	Blue	White	Overall
1	92%	100%	96%
2	92%	100%	96%
3	92%	100%	96%
4	92%	100%	96%
Over all success rate:	92%	100%	96%

Table 9.5: The classification success rate for the template matching.

Colour:	Blue	White
Mean score correct area:	0.99	0.99
Mean score wrong area:	0.988	0.97
Std. score correct area:	0.009	0.0004
Std. score wrong area:	0.0092	0.0015

Table 9.6: The mean matching score and the standard derivation (std.) of the the score. Note the data shown origins from 75 blue top covers and 25 white top covers.

are too similar, to obtain a stable and sufficient result, using NCC based template matching.

A possible solution to this would be to use another method, for matching the areas e.g. champher matching, where edge information of the area to be detected is used as template, and is matched against a distance transformed version of the input image.

Chapter 10

Calibration To FESTO CP Factory

The mobile positioning of LH6 is $\pm 10\ cm$. This means that in order for LH6 to execute a tasks which requires a precision bigger than the specified, a calibration which makes it possible to relate the base of the manipulator to a point in the world is needed.

This Chapter consists of a description of how the LH6 will be able to use a 2D camera to calibrate its position with respect to a static QR-code, when it arrives at the specified module at FESTO CP Fatory (see Section 5.2). It will in this project be use to place phone covers on a carrier at the module. In order to do the calibration two image coordinates, related to a static QR-code is localised (see Section 10.1) and after used to calculate the 3D position of the camera with respect to the QR-code using the method described in Chapter 8 (see Section 10.2).

10.1 Detection of QR Code

A QR-code is found in a monochrome image, with a high stability and robustness. One characteristic of a QR-code is the larger square boxes placed in three of the four corners of the QR-code. These boxes are used to find the QR-code while simultaneously finding the orientation of it. In Figure 10.1 one of the three corner-boxes are marked with green. The four corners gives the physical size of the

Figure 10.1: Example of a QR-code with the green highlighting one of the three corner boxes.

QR-code. From theses corners the center of the QR-code, c_{QR}, can be calculated and used as one of the wanted image coordinates:

$$c_{QR} = \left(\frac{x_{c1} + x_{c2} + x_{c3} + x_{c4}}{4}, \frac{y_{c1} + y_{c2} + x_{c3} + y_{c4}}{4} \right) \tag{10.1}$$

Where, x_{cn} and y_{cn} are the corners, of which one can be used to obtain the orientation coordinate. Based on c_{QR} and one of the corners, the transformation from the camera to the QR-code can be calculated using the method introduced in Section 8.

QR-codes possesses the extra feature of being capable of storing information such as links or text. Related to this project, the information can identify the module on which it is placed and then be used as a pre-condition for the LH6 skill to verify that the module is indeed the correct module.

10.2 Calibration at FESTO CP Factory

In order to obtain a flexible calibration method, the placement of the QR-Code is chosen to be on the FESTO CP Factory module and not on the carrier (see Figure 10.2). This means that, by assuming that the transformation from the camera to the QR-code, at FESTO CP Factory, ${}^{C}_{QR}T$, is obtained using the method introduced in Section 8, the transformation from the base of the manipulator to the QR-code, ${}^{B}_{QR}T$, can be obtained:

$${}^{B}_{QR}T = {}^{B}_{t}T \cdot {}^{t}_{C}T \cdot {}^{C}_{QR}T \tag{10.2}$$

Where ${}^{B}_{t}T$ is the transformation from the base to the tool of the manipulator, obtained using a known kinematic manipulator model, and ${}^{t}_{C}T$ is the transformation from the tool to the camera, which is obtained using the DLR calibration method for extrinsic camera parameters (see Section 7).

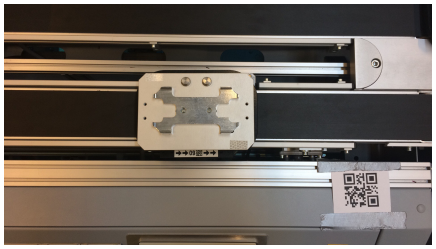


Figure 10.2: Placement of QR-Code.

However, the transformation needed in order to calibrate the LH6 with respect

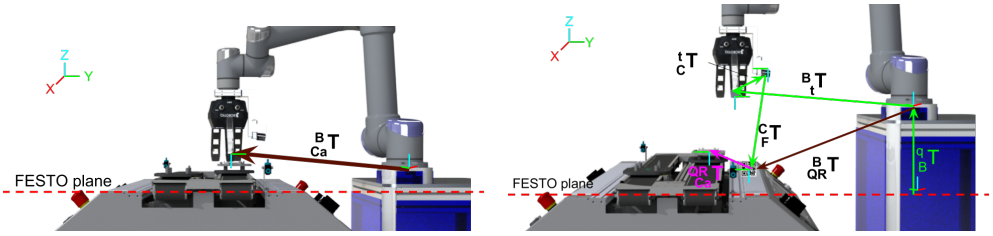
to the carrier, is the transformation from the base of the manipulator to the carrier, ${}^{B}_{Ca}T$. This transformation can be obtained using the transformation from the QR-code to the carrier, ${}^{QR}_{Ca}T$ and ${}^{B}_{QR}T$ (see Equation 10.3).

$$ {}^{B}_{Ca}T = {}^{B}_{QR}T \cdot {}^{QR}_{Ca}T \tag{10.3} $$

In Equation 10.3, ${}^{QR}_{Ca}T$ is a constant, found ether by manually measuring the transformation or by calculating it. By placing the tool of the manipulator on the carrier, the transformation ${}^{B}_{Ca}T$ can be obtained using the known kinematic model of the manipulator (see Figure 10.3a). This means that, ${}^{QR}_{Ca}T$ can be calculated using ${}^{B}_{Ca}T$ (see Equation 10.4). This process has to be done every time when teaching the manipulator a new position.

$$ {}^{QR}_{Ca}T = {}^{B}_{Ca}T \cdot \left({}^{B}_{QR}T \right)^{-1} \tag{10.4} $$

When ${}^{QR}_{Ca}T$ is obtained, it can be inserted in Equation 10.3 and ${}^{B}_{Ca}T$ can be calculated. Thereby the LH6 can be calibrated when it arrives at the FESTO CP Factory module, using a QR-code as shown in Figure 10.3.



(a) Teaching the transformation from manipulator base to the carrier.

(b) Using the method from Chapter 8 to calculate ${}^{B}_{QR}T$

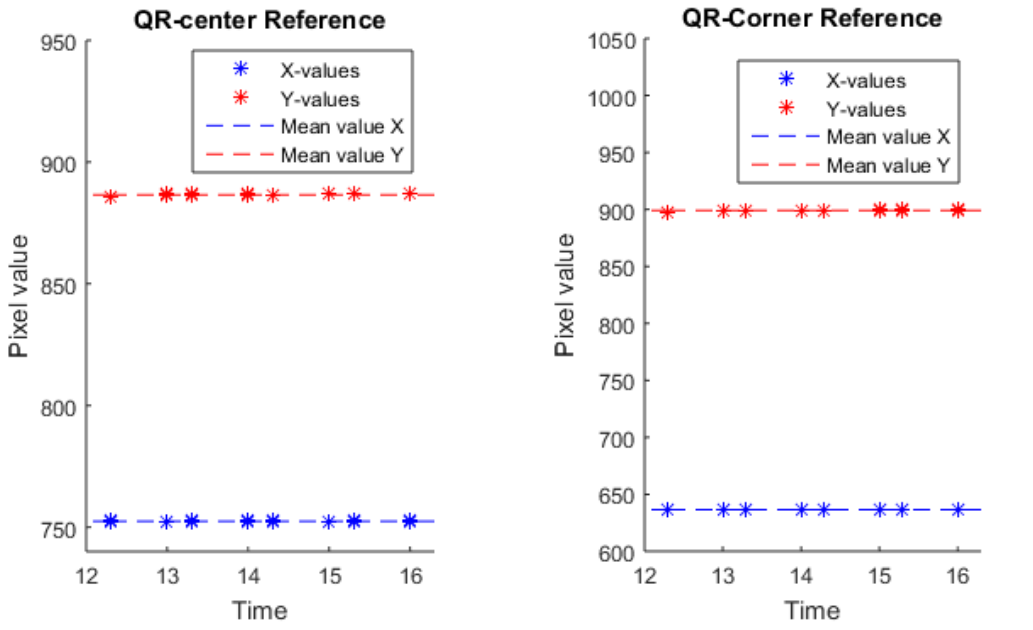
Figure 10.3: Illustration of how the ${}^{QR}_{Ca}T$ transformation is calculated.

10.3 Subresults for QR-code Calibration to FESTO Module

This section contains two tests. One to give an indication of how well the QR-code can be detected using the *zbar* library throughout a day (starting at noon) as well as testing the repeatability of the library. Another test is done to show what the influence of the variations from the first test has on the transformation from the base of the manipulator to the carrier.

The first test was done where a QR-code was located in an image 10 times. The

test was then carried out with 30 minutes intervals resulting in a total of 80 images. Both the QR-code and the camera was static as it was the influence of the light change during the day, that was being tested.



(a) Center. Mean=[752.44, 886.59] and STD=[0.30, 0.43].

(b) Corner. Mean=[636.53, 899.08], STD=[0.51, 0.57].

Figure 10.4: Pixel positions of the QR-code. Each star represents 10 tests.

The results in Figure 10.4 shows how the pixel positions fluctuate very little over time. This means that *zbar* gives a stable result, i.e. high repeatability, with a standard deviation of less than a pixel.

Using the method described in Chapter 8 the pixel positions from the first test are in the second test transformed into transformations, and multiplied with the constant transformation from the QR-code to the carrier:

$$\overset{C}{Ca}T = \overset{C}{QR}T \, \overset{QR}{Ca}T \tag{10.5}$$

Table 10.1 shows the position of the point above the carrier with all the points found in the first test. Though the LH6 was static throughout the test, the calculations still produced a standard variation of up to 0.7366 mm in the worst case. This could indicate that the small pixel-variations of the position of the QR-code does have an effect on the final position above the carrier. Since this test was done while the camera was statically placed, the precision of the camera which, was shown in

	Roll [rad]	**Pitch [rad]**	**Yaw [rad]**	**X [mm]**	**Y [mm]**	**Z [mm]**
Mean	-0.0643	0.0284	0.6827	186.4511	-24.8971	372.6995
STD	0.0001	0.0002	0.0030	0.4139	0.7366	0.00

Table 10.1: The mean and standard deviation of the transformation ${}^{C}_{Ca}T$.

Chapter 8 to vary depending on the position in the image, was not included in the results. If the camera had been moved during the test, the position of the QR-code and the resulting transformations may have accumulated to different results.

Chapter 11

Implementation

The following chapter consists of the implementations of the methodologeies from Chapter 6, 7, 8, 9, 10 and how they are interconnected through SBS.
The implementations can be divided up into four main parts consisting of the following:

MES modification
To enable the production of the blue and white covers, LH6 needs to be implemented in MES. This is done by adding the LH6 as a resource to MES (Section 11.1).

Communication between LH6 and FESTO CP Factory
In order to establish communication between LH6 and FESTO CP Factory, three programs are needed; one web-server for reading the MES database (Section 11.2). One ROS driver for connecting to the web-server and receiving information from the database (Section 11.2). One ROS driver for connection with the dedicated module (Section 11.3).

Object localisation and pose estimation
To detect phone covers and a QR-code and estimate their position in real world 3D coordinates, a ROS driver called **CameraPoseEstimation** is created (Section 11.4).

SBS implementation
This part consists of the skills implemented in SBS in order to enable intuitive programming of LH6 using the created drivers (Section 11.5 and 11.6).

Figure 11.1 illustrates the different sub-systems and their connections. Note, that the UR5, MiR100, Robotiq gripper and camera drivers are pre-existing drivers of SBS.

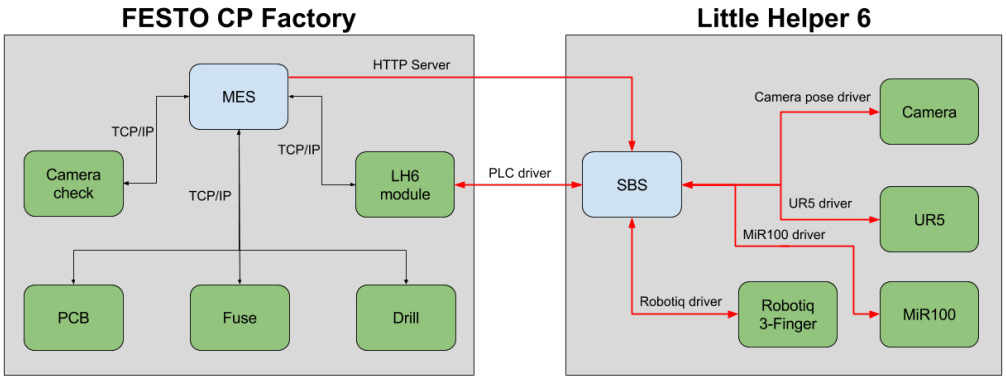


Figure 11.1: The components, processes and their communication. The red arrows are controllable connections. Blue boxes are software, and green boxes are hardware.

11.1 Implementation of LH6 in MES

This section describes how LH6 is integrated into FESTO CP Factory. This is done based on the communication described in Chapter 6.

11.1.1 MES Integration

To successfully integrate a special priority order into FESTO CP Factory, the order needs to be implemented in MES. This is done accordingly to how MES is build up and how the black phones are implemented. For the blue and white dummy smart-phones, to be implemented according to what is described in Chapter 2, the following four things are needed: resource, parts, operations and work plan.

Resource

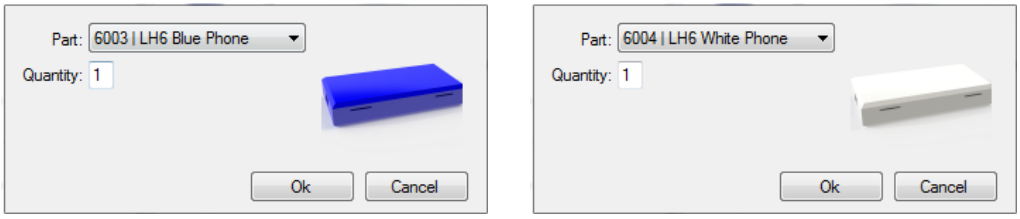
The resources of FESTO CP Factory are the machines that do the operations, such as the drilling station and KUKA module for PCB and fuses. A new resource is needed since none of the existing will allow a bottom and top covers placement with LH6. Thus an unused module, namely *CP BASIC 3*, is chosen (shown in Figure 5.1). The AppModul program described in Section 6.2 is applied which changes the name of the module to *CP Factory LH6* and gives the IP address 172.20.7.1. Thereby MES is able to assign operations to the module, as seen in Figure 11.2.

Parts

The dummy smart-phones are implemented in MES as parts with different configurations. Each of the different parts have a unique number (PartNumber) which is used to distinguish one from the others. Furthermore, the PartNumber's are

Figure 11.2: All the resources of MES, including LH6's dedicated module, CP Factory LH6, with ID number 102 (highlighted in blue).

used to assign the different orders to a specific work plan. For this project the number 6003 and 6004 were assigned for the blue and white dummy smart-phone, respectively as shown in Figure 11.3.



(a) Blue dummy smart-phone. **(b)** White dummy smart-phone.

Figure 11.3: The order window in MES, where the user selects which dummy smart-phone is wanted.

Operations

Operations are the different actions within FEESTO CP Factory such as drilling. There can be multiple operations such as drilling left, drilling right ect. on a single resource. For this project there will be three new operations added. The operations

are given the unique operation numbers 952, 951, 950 and are all used on the *CP Factory LH6* module.

952: The first operation has one job, and that is to assign a carrier to the order. When this is done the operation is over.

951: Here the PLC calls for LH6 and the carrier waits for LH6 to come and place a bottom cover. When it have received a confirmation from LH6, the operation is done.

950: The last operation the PLC calls for LH6 and the carrier waits for LH6 to come and place a bottom cover. When it have received a confirmation from LH6, the operation is done.

These three operation are not colour dependent, thus they will be the same for the blue and white dummy smart-phone.

Work Plan

A work plan combines the parts and operations with the rest of FESTO CP Factory in a production recipe. A new work plan was created to encapsulate the new parts, operations and resource. This work plan (with the work plan number 17) is dedicated for both the blue and white dummy smart-phones, as the production recipe is not colour dependant.

The reason for this is that the only colour dependent action is done by LH6, where it is told what colour to fetch which is given through the web-server and taught through SBS. The work plan is illustrated in Figure 11.4.

11.1.2 Network Setup

The integration of LH6 and FESTO CP Factory is dependent on a solid network setup. Since FESTO CP Factory and LH6 runs on two separate networks, it is a requirement of the computer running SBS to be on both networks. In Figure 11.5 the network setup is shown.

11.2 ROS Driver and HTTP Server Between SBS and MES

In order for the LH6 to know when an order is placed, it is necessary for it to read the database of the MES, where all the orders are stored.
When the MES server program, described in Section 6.1, is launched, it will first try to connect to the accdb database. In Listing 11.1 it is shown that calling the method *LoadConnection()* utilise the variable **connection** with a connection string to access the MES database.

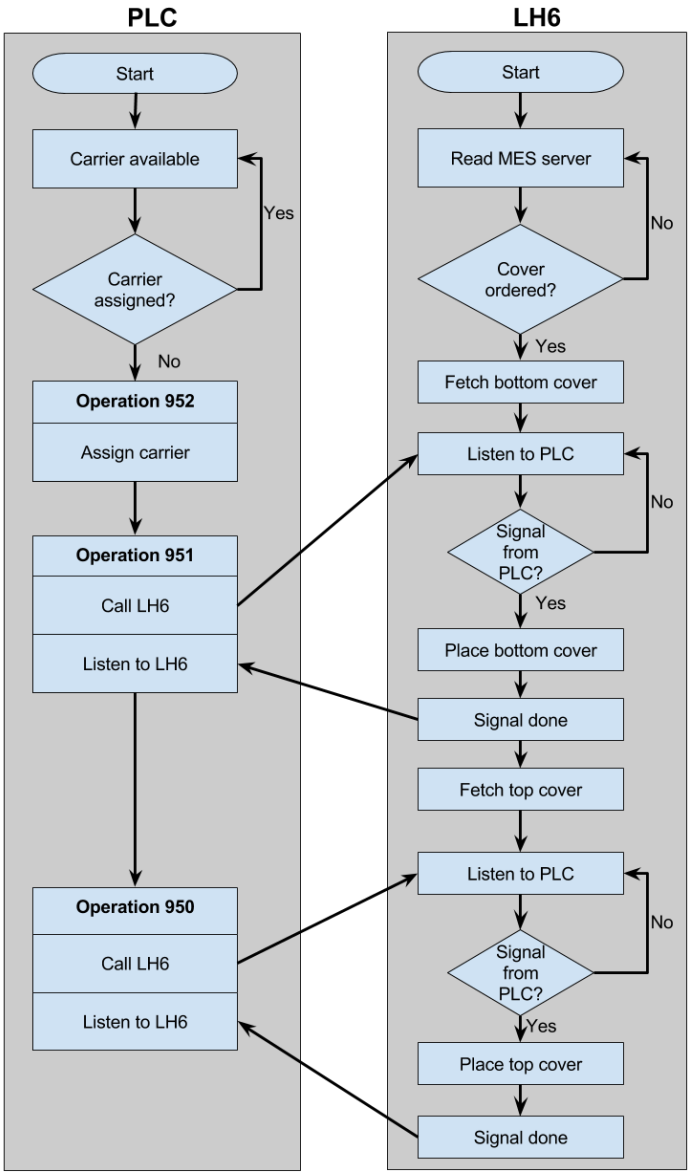


Figure 11.4: Flowchart of work plan 17.

```
LoadConnection()
  connection.ConnectionString =
     @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" +
     mesDatabasePath + @";Persist Security Info = False;";
  if connection is open
    return "MES database is open"
  else
    return "failed to open MES database"
```

Listing 11.1: Code snippet of the method LoadConnection(). This method tries to access the FESTO CP Factory's database.

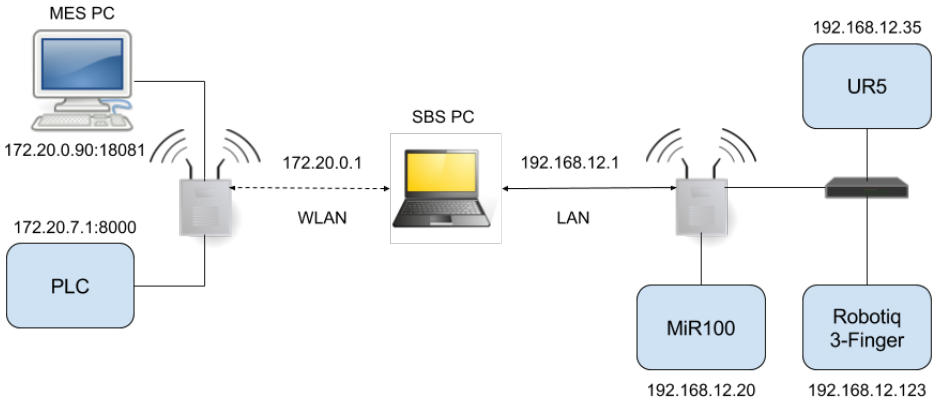


Figure 11.5: Network setup of LH6 and FESTO CP Factory.

When connection has been established between the database and the MES web-server, the program will create a server, on the local computers IP-address through port 18081. When a client connection, the program will read the MES database and parse all the data related to new orders into local variables, this can be seen in Listing 11.2.

```
reader = open MES database table

OrderNumber = reader.GetInt32(0),
Start = reader.GetDateTime(4).ToString(),
End = reader.GetDateTime(5).ToString(),
WorkPlanNumber = reader.GetInt32(6),
ResourceID = reader.GetInt16(10),
OperationNumber = reader.GetInt32(11),
PartNumber = reader.GetInt32(13),
Error = reader.GetBoolean(15)
...
```

Listing 11.2: Data from MES database assigned to local variables.

After all the data has been parsed into the local variables, they are used to create a JSON format (shown in Listing 6.1) which is served to the connected client over the HTTP connection.

A ROS driver is created that can connect to the MES web-server together with a corresponding skill in SBS. The skill has the purpose of stalling the SBS task execution until a new order is received. This is done by calling the ROS driver at a frequency of 1 Hz and not proceeding execution before the driver returns that a new order has arrived.

11.3 ROS Driver and TCP/IP Server Between SBS and LH6 Module

Similar to the web-server in Section 11.2, a TCP/IP server is created to communicate with the PLC in the *CP Factory LH6* module. This server receives information from the PLC when a carrier is available and waiting for the LH6. When the PLC connects to the TCP/IP server the server respond with true or false of whether the LH6 has finished its operation, i.e. placed the part. By making a service call, LH6 signals the TCP/IP server when a phone cover is placed.

11.4 ROS Driver CameraPoseEstimation

The CameraPoseEstimation driver has the purpose of returning the position of the object in an image (e.g. blue or white cover), represented as the transformation from the camera origin to the object. For this to be possible two inputs are necessary for the driver; an image containing the object and the transformation between the camera and the plane on which the object lies. These inputs are necessary in order to use the pose estimation explained in Chapter 8 and object detection explained in Chapter 9 or 10. The transformation from the plane to the camera is obtained partly from the kinematic model of the manipulator and partly when the robot is being taught. The driver consists of two main parts, for these inputs, listed in the following:

1. Object detection
This part consists of an implementation of the object detection methods described in Chapter 9 and Chapter 10. The output of this part is a position of a given object in 2D image coordinates, which is given as an input for the next part (see Figure 11.6).

2. Pose estimation
This part consists of an implementation of the pose estimation method described Chapter 8. This method calculates and returns a transformation from the camera to the object based on the 2D image position (see Figure 11.6).

The an overall illustration of the CameraPoseEstimation driver shown in Figure 11.6 and sketched in Listing 11.3. Here the input is the mentioned transformation and the object that is to be found, and the output is the transformation from the camera to the object.

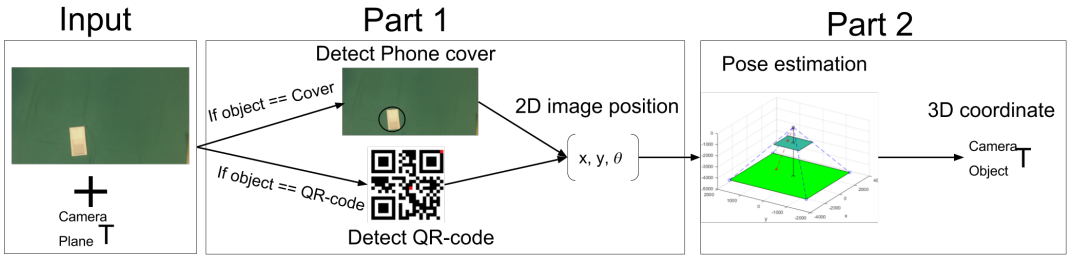


Figure 11.6: The basic parts of the CameraPoseEstimation driver.

```
function camera2phone(camera2plane, object)
  image = read undistorted image from camera topic
  if object == QR-code
    image = ConvertToGrayScale(image)
    [objPosition, objRotation] = QR_finder(image)
  else if object == phone cover
    binaryImage = getBinaryImage(image)
    phoneCovers = representBlobs(binaryImage)
    phoneCovers.classes = classify(phoneCovers)
    freeCover = findFreeCover(phoneCovers)

    if freeCover.class == bottomBlue || bottomWhite
      objOrientation = topCoverOrientation(freeCover, image)
    else if freeCover.class == topBlue || topWhite
      objOrientation = freeCover.orientation

    ray = ConstructRay(objPosition)
    intersection = CalculateRayIntersection(ray)

    camera2phone = transformation(RotationAroundZ, 0, 0,
                                  intersection.x,
                                  intersection.y,
                                  intersection.z)
                                  // ZYX-Euler Angles
    return camera2phone
```

Listing 11.3: Process of CameraPoseEstimatino driver

11.5 Pick Skill

In order to make the LH6 pick up phone covers, a skill called *PickPhoneSkill* is implemented in SBS. This skill calls the CameraPoseEstimation driver, and thereby obtains a transformation from the camera to a phone cover. Based on this transformation SBS then simply multiplies the transformation between the manipulator

base and tool, tool and camera and lastly the newly create transformation between camera and phone cover, this is done according to Figure 11.7.

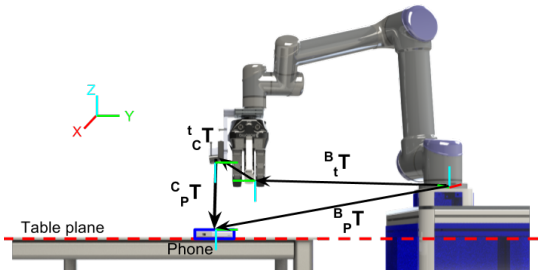


Figure 11.7: Transformation from the base of the robot to the phone cover.

Before the manipulator can pick up the cover, some preconditions has to be fulfilled; there should be a cover of an ordered colour and type and the cover should be available and free. If the preconditions are not fulfilled the manipulator will search for a cover by moving its first joint 5° to clockwise, and call the driver again. This will be repeated until the manipulator finds a free cover, or after 10 tries with no success. If a cover of the ordered class is detected the manipulator will execute the skill and pick up the given cover. After the execution some post-conditions can be implemented e.g. checking if there are one less cover of a certain class after performing the execution (see Figure 11.8). The only parameter specified when teaching this skill, is the transformation from the plane to the camera. This is obtained by moving the manipulator to the plane where the phone covers lie and store that position.

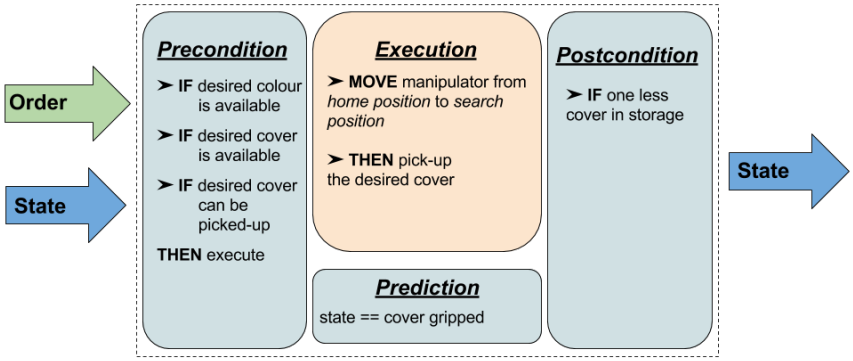


Figure 11.8: An example of the pick skill.

11.6 Place Skill

When the LH6 has picked op a cover, it should place it on the carrier at FESTO CP Factory. Therefore, a skill called *PlacePhoneSkill* is implemented. This skill has the preconditions that the carrier is available and the QR-code is visible (see Figure 11.9). In order to place the phone cover on the carrier, a transformation from the QR-code to the carrier is necessary. This transformation is constant and found when teaching the skill. By moving the manipulator to the carrier, the transformation can be obtained using the method described in Chapter 10. Beside this transformation the plane where the QR-code is placed is taught by moving the manipulator to the QR-code when teaching the skill. One example of a postconditions for this skill could be to check if the placement of the phone cover is successful using vision. Figure 11.10 illustrates the process of teaching the *PlacePhoneSkill*

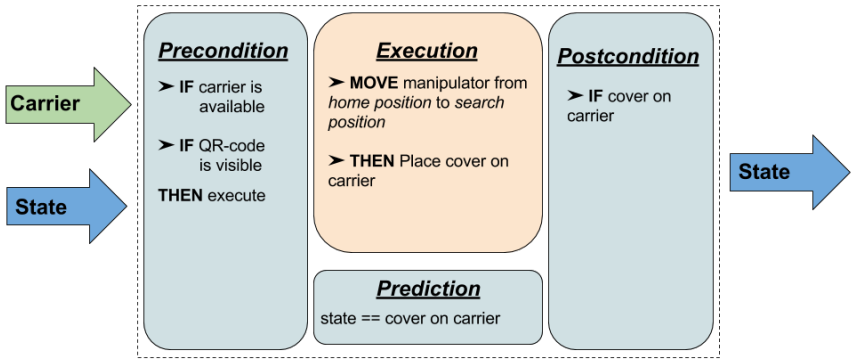
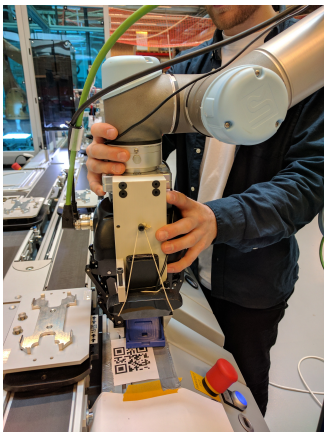
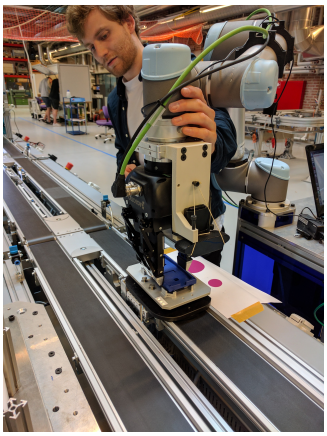


Figure 11.9: An example of the place skill.



(a) Teaching the height of the QR-code plane.



(b) Teaching the position of the cover with respect to the QR-code.

Figure 11.10: The teaching procedures of the *PlacePhoneSkill*.

Chapter 12

Final Test

The final test was carried out to test the performance of the system in the context of special priority orders at FESTO CP Factory, described in Section 5.4. This chapter gives a description of the test setup along with its results. The test is validated with success/fail criteria.

12.1 Test Setup

The test was taught in SBS on a computer running the network setup from Section 11.1.2. Six different tasks were created which combined, would carry out the test. The tasks created for the test were the following, where * indicates that there is a corresponding task for white covers:

task_0_start_pos
: The initial task was created to drive LH6 to an initial start position (0 in Figure 12.2), along with making sure the manipulator where orientated correctly.

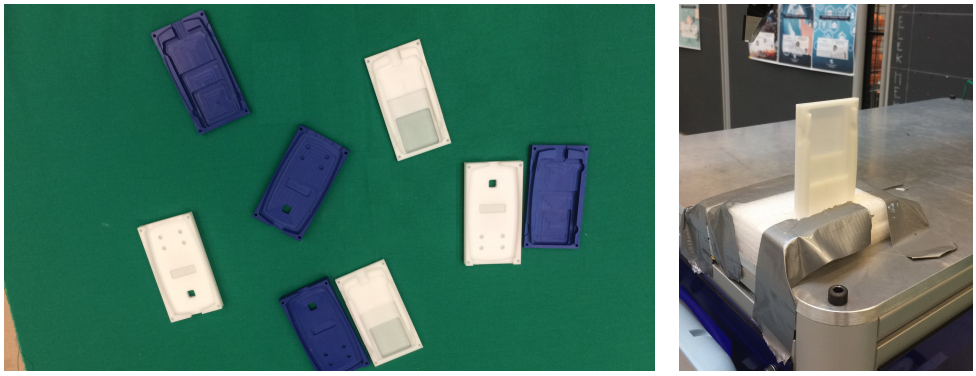
task_1_w_bottom_blue*
: LH6 will in the start of this task read the MES database, and when a blue special order is placed, LH6 will drive to the storage facility (1 in Figure 12.2). Here LH6 will find the correct bottom cover on a table, where two of each type of cover is located. Note, that the covers are placed such that only one of each cover is accessible (See Figure 12.1a). When a bottom cover is picked up the task is done.

task_2_FESTO
: LH6 will start with driving to the *CP Factory LH6* module (2 in Figure 12.2). When LH6 arrives it will ask the PLC if the carrier is ready. When a ready signal is given, LH6 will place the bottom cover on the carrier and send a confirmation to the PLC.

task_3_w_top_blue*
LH6 will then drive back to the storage facility (3 in Figure 12.2) and pick up a top cover.

task_4_w_place_top_on_lh6
Because the top cover is facing upwards on the storage table, it needs to be flipped in order to place it on the bottom cover. Here LH6 will put the top cover in a vertical stand for easy picking at the *CP Factory LH6* module, which is shown in Figure 12.1b.

task_5_FESTO
At last, LH6 will drive back to the *CP Factory LH6* module (5 in Figure 12.2), pick up the top cover from the stand and listen for the carrier to be ready. When the carrier is ready, LH6 will place the top cover on the bottom cover and signal to the PLC that the cover has been placed.



(a) The storage facility.

(b) Top cover stand on LH6.

Figure 12.1: Storage facility and top cover stand

The assembled phone will then move along the FESTO CP Factory to the module for manual picking of the assembled phones.

The majority of the skills used in this test were skills created in this project, besides the pre-existing DriveTo and a few MoveTo skills. The manipulator speed were in all skills set to 30%, which was assumed to be a good speed for this environment containing humans.
In Figure 12.2, the map layout of the test setup is shown.

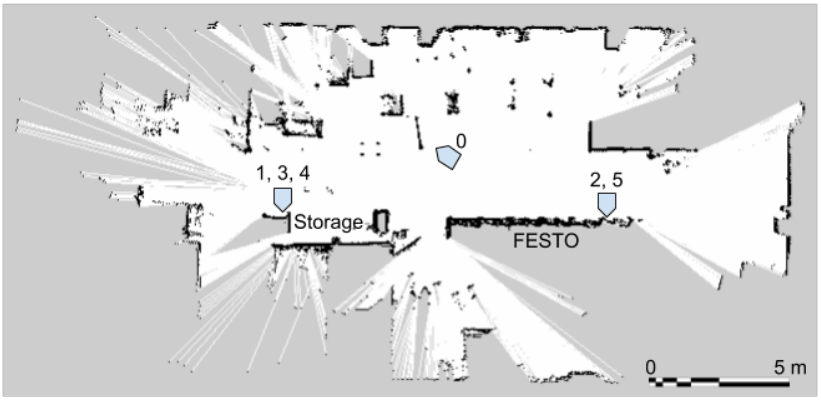


Figure 12.2: The map LH6 has of the test environment. The numbers 0 through 5 indicates the place, where the task with the same number, is carried out.

12.2 Test Results

The final test was carried out 20 times, where the blue phones were the first 10 test and white phone the last 10. The test was evaluated on seven parameters:

MES: LH6 successfully read and interpret the web-server from MES.

WB (Warehouse Bottom): LH6 is able to drive to the storage facility, detect and pick up the right coloured bottom cover.

FB (FESTO Bottom): Successfully drive to the *CP Factory LH6* module, calibrate the offset and place the bottom cover on the carrier.

PLC: LH6 is able to wait for the signal that is telling the carrier is ready. Furthermore, send a signal to the PLC after the LH6 is done with its operation.

WT (Warehouse Top): LH6 is able to drive to the storage facility again, find the correct coloured top cover, including the correct orientation. Then pick it up and place it in the stand on the LH6 table.

FT (FESTO Top): Drive to the *CP Factory LH6* module, calibrate the position and place the top cover correctly.

Time: The overall time in seconds from when an order is placed to when the PLC receives the last signal from LH6.

Each parameter was evaluated on a success/fail criteria. If one parameter fails, the trial is considered a fail since the next operations will not be possible to carry out. The FESTO CP Factory products were originally black dummy phones and not blue and white, which was a problem for the backlit orientation check in the

Number	Colour	MES	WB	FB	PLC	WT	FT	PLC	Time [s]	Overall
1	Blue	✓	✓	✓	✓	✓	✓	✓	459	Success
2	Blue	✓	✓	✓	✓	✓	✓	✓	450	Success
3	Blue	✓	✓	✓	✓	✓	✓	✓	445	Success
4	Blue	✓	✓	-	✗	-	-	-	-	Fail
5	Blue	✓	✓	✓	✓	✓	✓	✓	424	Success
6	Blue	✓	✓	✓	✓	✓	✓	✓	490	Success
7	Blue	✓	✗	-	-	-	-	-	-	Fail
8	Blue	✓	✓	✓	✓	✓	✓	✓	490	Success
9	Blue	✓	✓	✓	✓	✓	✓	✓	461	Success
10	Blue	✓	✓	✓	✓	✓	-	✗	-	Fail
11	White	✓	✓	✓	✓	✓	✓	✓	451	Success
12	White	✓	✓	✓	✓	✓	✓	✓	409	Success
13	White	✓	✓	✓	✓	✓	✓	✓	375	Success
14	White	✓	✓	-	✗	-	-	-	-	Fail
15	White	✓	✓	✓	✓	✓	✗	-	-	Fail
16	White	✓	✓	✓	✓	✓	✗	-	-	Fail
17	White	✓	✓	-	✗	-	-	-	-	Fail
18	White	✓	✓	✓	✓	✓	✓	✓	458	Success
19	White	✓	✓	✓	✓	✓	✗	-	-	Fail
20	White	✓	✓	✓	✓	✓	✓	✓	510	Success
Fails	-	0	1	0	3	0	3	1	-	8
Meas.	-	20	20	16	19	16	15	13	12	20
Success (%)	-	100	95	100	84.2	100	80	92.3	avg. 452	60

Table 12.1: The result from 20 tests with 10 test of each colour. The mark ✓ indicates success, ✗ failure and **-** not recorded.

KUKA module. Therefore, for the test with white dummy smart-phones, the covers were replaced with black covers after LH6 successfully placed the white bottom cover on the carrier. Because the KUKA module is outside of this project scope, this will not be counted as an error.

In Table 12.1 the results of the final test are shown. The test yielded an overall success rate of 60% and an average cycle time of 452 seconds (7 min. 32 sec.). In a task like the one carried out here, good precession is needed when teaching. This is because the operator of SBS is required to physically move the manipulator to the cover placement position on the carrier. Therefore, it is not possible to know if bad teaching could cause imprecision when a cover is place in a carrier.

Chapter 13

Discussion

To determine the stability of the implementation of special priority orders on FESTO CP Factory with LH6, a final test of 20 trials were carried out.

In total 8 of the 20 trials failed, which gives an overall success rate of 60%. The implementation failures could be traced down to two issues; PLC communication and FESTO top placement.
The PLC communication failed in total four times, where LH6 could not receive the data the PLC sent when the carrier arrived. This issue could be because the FESTO CP Factory serves as learning equipment, thus multiply people were using it at the same time. Furthermore, the router which the PLC and LH6 was connected to was unstable and at times unable to maintain a stable connection. These factors were assumed to be the main reason of why the PLC communication failed.
Three times the placement of the top covers failed. These fails were because the manipulator was unable to grip the top cover from its stand (the stand is shown in Figure 12.1b). The stand was made of Styrofoam and it could be noticed over time that the quality and fit of it decreased. Moreover, it can also be seen from Table 12.1 that the first error occurred on trial 15, with the two other on 16 and 19. Furthermore the LH6 did not always grab the cover around the center causing the cover not to be correctly placed in the stand. A combination of these two circumstances were assumed to be the cause of this error.
The only error left is trial 7 at the storage facility when fetching bottom covers. Here the classifier classified a blue top cover as a blue bottom. The cause of this, was a bad segmentation, where reflected light containing a different *hue* value, created holes in the segmentation of the top cover. A way to avoid this classification error could be to add a morphology filter after the segmentation.

When the test was first taught, LH6 was parked parallel with the FESTO CP Factory. This turned out to be a problem for the MiR100's navigation system, since

it used minutes on placing itself correctly. Therefore, the DriveTo skill was taught again. Parking the LH6 with its front towards FESTO CP Factory solved this problem.

Before the final test, it was a concern that LH6 would not have reached the *Festo CP LH6* module before the carrier had gone a full round on FESTO CP Factory (see Sectionsec:designReq). However, for most of the trials, LH6 arrived before the carrier did. This is partly due to the implementation of the MES database communication where, more often than not, LH6 would get the order before a carrier was assigned. On the second round the bottom cover is going through the rest of the process' on FESTO CP Factory, meaning a lap time will take 4 min. 3 sec. instead of 2 min. and 20. sec., giving LH6 more than necessary time to fetch the top cover.

The time measured in the final test was from when an order was placed in MES, to when the carrier was released from the module. The highest measured time (510 sec.) for a production cycle was below the set requirement of 560 sec. The difference between the lowest measured time (375 sec.) and the highest was significant, indicating that the production time is unreliable. Since the LH6 was often waiting for the carrier, it can be assumed that this unreliable production time was due to the random time it takes for an order to be assigned to a carrier.

During the teaching of the PlacePhoneSkill it was discovered that the kinematic model of the manipulator did not allow for intuitive and precise teaching. This posed a problem when the especially the top cover had to be placed on the bottom cover with high precision.

Chapter 14

Future Work

In this chapter, subjects for future possibilities for this project will be discussed.
This project focused on picking phone covers in the storage facility where one of the challenges was the grippers capability to grasp objects. The top covers had to be flipped in order to be placed on a carrier. Flipping the cover might not be the best solution and therefore future optimisation could include considering another gripper and/or looking to classifying the top covers facing the other way.

Currently the LH6 is not a part of the MES. However, the LH6 and future LH's should be directly part of the MES, which would make the interconnection possible such as error handling e.g. the MES will be notified if the LH6 fails at performing its order. Furthermore, by adding the LH6 in the MES, there will not be a need for the program which enables the web-server to read from the database. Moreover, the TCP/IP connection between LH6 and the PLC, will possibly not be needed or atleast more stable.
Currently the template matching and orientation detection of top covers does not fulfil the requirement. Therefore, other methods could be considered instead of NCC based template matching.
The implemented pick and place skill does currently not contain any postcondition check. This should in the future be implemented such that LH6 would not know if it succeeded its operation.
Furthermore, in order to make the image processing more stable, a controlled light setup could be looked into.

Chapter 15

Conclusion

The focus of this project was to increase the product variation at FESTO CP Factory by adding the possibility of ordering special priority order with low occurrence. It was set to be solved with the LH6 which was analysed along with FESTO CP Factory. Two phone covers (blue and white) was chosen to be the special priority orders, since they were not already implemented in MES. Image processing challenges was identified and analysed to solve the task. Afterwards a problem statement was derived: *How can blue and white smart-phone dummies be integrated into the FESTO CP Factory using LH6 with a vision system and SBS?*, which identified the goals of the project and thus a requirement specification could be established. Because of time constrains, a delimitation of the requirements and project were made. This resulted in a final requirement list, as seen in Table 15.1.

The Requirement **H.1**, Requirement **H.2** and Requirement **H.3** was achieved by mounting a Logitech HD C920 Webcam and programming a static positions for the manipulator. The software used for controlling LH6 was SBS, thus achieving Requirement **S.1**. The time of the algorithm for localising the covers was not measured and thus Requirement **S.3** was undetermined. The precision of the position estimation was evaluated to be $\pm$16 mm, and did not fulfil the requirement of $\pm$5 mm, and the angle precision requirement was undetermined. Thus Requirement **S.4** was not met. With the utilisation of the web-server, a generic communication was establish, thus achieving Requirement **S.5**. Due to a classification success rate above 99% the Requirement **S.8** was met. The Requirement **S.9** was not reached because of the template matching for finding the orientation of the top cover. The final test showed a stability of 60% meaning that Requirement **O.2** is not met. The storage facility was utilised with a green static background according to Requirement **O.3**. The required operation time was set to 560 seconds. From the final test, the slowest overall production time was 510 sec. and the average time was 452 sec., thus Requirement **O.4** was achieved. Because of time constrains,

	Description	Value	Unit	Success
H.1	Static camera position above table	-	-	✓
H.1	Camera view angle with table	90	degrees	✓
H.2	Camera is RGB and minimum resolution	1920x1080	pixels	✓
H.3	Minimum field of view	32.38	degrees	✓
S.1	Software framework	SBS	-	✓
S.3	Localisation time	1	s	?
S.4	Pose position estimation precision	$\pm$ 5	mm	✗
S.4	Pose angle estimation precision	$\pm$ 5	degrees	?
S.5	Communication interface between MES and SBS	-	-	✓
S.8	Object classification	99	%	✓
S.9	Top cover orientation	99	%	✗
O.2	Stability of implementation	95	%	✗
O.3	Storage facility with static background	-	-	✓
O.4	Operation time for complete production cycle	560	seconds	✓
O.5	Error handling for missing part	-	-	✗
O.7	Storage facility maximum distance to FESTO	10	m	✓
O.8	LH6 placement distance	10	m	✓

Table 15.1: Table of final requirements with review.

error handling for missing part at the storage facility was not implemented and thus not achieving Requirement **O.5**. Requirement **O.7** and Requirement **O.8** were fulfilled due to vital WiFi connection and travel time for LH6. In the final test map (in Figure 12.2) it can be seen that the LH6 should never travel further away than 10 m. Furthermore, it can be seen that the storage facility is placed approx. 7 m. away from the nearest FESTO module. All the requirements achieved and failed can be seen in Table 15.1

The LH6 solving the special priory cover in the final test with 60% success rate, raised a few issues. One was the unstable TCP/IP connection with the PLC and another was the Styrofoam loosing its fit over time. These two issues could be solved with more time. A more stable connection could be investigated and a more solid material than Styrofoam should be used.
Another concern was the precision of teaching. During the final test, the top covers were partly misplaced. It is assumed that this was due to a combination of bad teaching and the imprecision of the pose estimation method introduced in Section 8.2. Nonetheless, it was shown that LH6 using SBS and vision has the potential to be used for an expansion of the product variety of the FESTO CP Factory.

Bibliography

Andersen, R. S., J. S. Damgaard, O. Madsen, and T. B. Moeslund (2013a). "Fast calibration of industrial mobile robots to workstations using QR codes". In: *IEEE ISR 2013*, pp. 1–6. DOI: 10.1109/ISR.2013.6695636.

Andersen, Rasmus Echholdt, David Cerny, Emil Blixt Hansen, Steffen Madsen, and Biranavan Pulendralingam (2016). *Autonomous Industrial Mobile Manipulator in Smart Production*. Research rep. Aalborg University.

Andersen, Rasmus S., Casper Schou, Jens S. Damgaard, Ole Madsen, and Thomas B. Moeslund (2013b). "Human Assisted Computer Vision on Industrial Mobile Robots". In: *Proceedings of the 1st AAU Workshop on Human Centered Robots*.

Andersen, Rasmus Skovgaard, Ole Madsen, and Thomas B. Moeslund (2014). *Hand-Eye Calibration of Depth Cameras based on Planar Surfaces*.

Bøgh, Simon, Oluf Skov Nielsen, Mikkel Rath Pedersen, Volker Kruger, and Ole Madsen (2012). *Does your Robot have Skills?* Tech. rep. Aalborg University. URL: http://vbn.aau.dk/files/69945674/ISR_2012_Paper_ID_1158_FINAL.pdf.

Corke, Peter (2011). *Robotics, Vision and Control, Fundamental algorithms in Matlab*. Springer Tracts in Advanced Robotics.

Europe (2017). *Robotics 2020, Multi-Annual Roadmap*. SPARC. URL: https://www.eu-robotics.net/cms/upload/topic_groups/H2020_Robotics_Multi-Annual_Roadmap_ICT-2017B.pdf.

Festo-Didactic-SE (2017). *Industry 4.0 - Qualification for the factory of the future*.

Freeman, H. and R. Shapira (1975). "Determining the Minimum-area Encasing Rectangle for an Arbitrary Closed Curve". In: *Commun. ACM* 18.7, pp. 409–413. ISSN: 0001-0782. DOI: 10.1145/360881.360919. URL: http://doi.acm.org/10.1145/360881.360919.

Hartley, Richard and Andrew Zisserman (2003). *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press.

Hvilshøj, Mads, Simon Bøgh, Ole Madsen, and Morten Kristiansen (2010). "Calibration Techniques for Industrial Mobile Manipulators: Theoretical Configurations and Best Practices". In: *Proceedings for the joint conference of ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*. VDE Verlag GMBH. ISBN: 978-3-8007-3273-9.

Logitech (2017). URL: http://www.logitech.com/en-us/product/hd-pro-webcam-c920.

Madsen, Ole (2016). "AAU Smart Production". In:

MathWorks (2014). URL: https://se.mathworks.com/help/vision/ref/cameraparameters-class.html.

Moeslund, Thomas B. (2012). *Introduction to Video and Image Processing Building Real Systems and Applications*. Springer.

Pedersen, Mikke Rath (2011). "Integration of the KUKA Light Weight Robot in a Mobile Manipulator". MA thesis. Department of Mechanical and Manufacturing Engineering, Aalborg University.

Pedersen, Mikkel Rath, Lazaros Nalpantidis, Rasmus S. Andersen, Casper Schou, Simon Bøgh, Volker Kruger, and Ole Madsen (2016). "Robot skills for manufacturing: From concept to industrial deployment". In: *Robotics and Computer-Integrated Manufacturing* 37, pp. 282–291.

Russmann, Michael, Markus Lorenz, Philipp Gerbert, Manuela Waldner, Jan Justus, Pascal Engel, and Michael Harnisch (2015). *Indusry 4.0 - The Futureof Production and Growth in Manufacturing Industries*. Tech. rep.

Shiu, Yiu Cheung and Shaheen Ahmad (1989). "Calibration of Wrist-Mounted Robotic Sensors by Solving Homogeneous Transform Equations the Form AX = XB". In: *IEEE*.

Strobl, Klaus H. and Gerd Hirzinger (2006). "Optimal Hand-Eye Calibration". In: *IEEE/RSJ*.

Wikipedia (2017). *HSV Image*. URL: https://en.wikipedia.org/wiki/HSL_and_HSV#/media/File:HSV_color_solid_cylinder_alpha_lowgamma.png.

World (2016). *robrobot og roboturobot- samarbejdende robotter*. Tech. rep. Danish standard foundation og international teknisk specifikation.

Appendix A

Setup Guide for Running LH6 With FESTO CP Factory

To enable the LH6 to run with FESTO CP Factory, some programs and certain setup is needed. This appendix consists of a guide to set up the LH6, and enable it to work with FESTO CP Factory. It is assumed that the user already has SBS and ROS Indigo installed. Together with those, the following software dependencies are also needed; usb_cam and zbar.
Throughout the guide there will be referred to two computers, the one running MES (MES PC) and the computer located on LH6 running SBS (SBS PC).

The following list, provides a step by step guide on setting LH6 and FESTO CP Factory up:

1. **Turn equipment**
   Turn on MiR100, UR5, Robotiq 3-Finger, FESTO CP Factory and the two computers.

2. **Connect the SBS PC network**
   Connect the SBS PC to MiR100 by Ethernet and to FESTO CP Factory by WiFi. The network should be setup as showed in Section 11.1.2. The SBS PC IP-address to FESTO CP Factory WiFi should have a static IP-address *172.20.0.102* and the network mask set to *255.254.0.0*.

3. **Upload the PLC program**
   The PLC program (called **_BASIC_3** in Codesys) should be uploaded to the *BASIC 3* module on FESTO CP Factory.

4. **Start MES and MES web-server**
   Start MES on MES PC. Afterwards, execute **mesAccdbProgram.exe** on MES

PC to start the web-server. The web-server is available on *172.20.0.90:18081* through a browser.

5. Start the drivers on SBS PC

Each of the following commands should be run in their own terminal:

- Start ROScore:
  roscore
- Start the MiR100 driver:
  rosrun mir mir_driver.py
- Start UR5 drive:
  roslaunch ur_bringup ur5 _and_proxy_bringup.launch
- Start Robotiq gripper driver nr. 1:
  rosrun robotiq_s_model_control SModelTcpNode.py 192.168.12.123
- Start Robotiq gripper driver nr. 2:
  rosrun rq3_proxy rq3_proxy_v3.py
- Start USB-cam driver (here the settings of the camera should be as in Table 7.1 and located in a launch file called usb_cam-external.launch):
  roslaunch usb_cam usb_cam-external.launch
- Start image processing driver:
  ROS_NAMESPACE=usb_cam rosrun image_proc image_proc
- Start camera pose driver:
  rosrun CameraPoseEstimation CameraPoseEstimation_node
- Start MES web-server driver:
  rosrun websocket_mes web_socket_server.py
- Start PLC driver:
  rosrun tcp_driver plc_communicate.py

6. Start SBS

Now start SBS on SBS PC with the following command in a new terminal:
rosrun Skill_Based_System SBS -r lh6

SBS is now able to perform actions together with FESTO CP Factory. To utilise LH6 with FESTO CP Factory the same way as in the final test, see Section 11.1.