Melina Nygard Karlsen
Marianne Nerland
Thomas Hellstrøm Olsen

Automated Paint Bucket Handling: Robotic Solution for Picking and Stacking

Graduate thesis in BIAIS
Supervisor: Adam Leon Kleppe
Co-supervisor: Paul Steffen Kleppe
May 2024

Melina Nygard Karlsen
Marianne Nerland
Thomas Hellstrøm Olsen

Automated Paint Bucket Handling: Robotic Solution for Picking and Stacking

Graduate thesis in BIAIS
Supervisor: Adam Leon Kleppe
Co-supervisor: Paul Steffen Kleppe
May 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

Preface

This document presents our bachelor's project in Automation and Intelligent Systems at NTNU Ålesund, and was prepared by Melina Nygard Karlsen, Thomas Hellstrøm Olsen and Marianne Nerland.

This project has been incredibly interesting and educational to work on, as it addresses a relevant issue in the industry. The project focuses on how to pick and place paint buckets using a robot and addresses all potential problems associated with this task.

Acknowledgement

We would like to thank Adam Leon Kleppe and Paul Steffen Kleppe for their contribution as our supervisors during this project.

We also extend our gratitude to Solwr for providing the project and lending equipment, and a special thanks to our client Olivier Roulet-Dubonnet and our contact person in Ålesund, Rasmus Nes Tjørstad, who have been very helpful throughout this project.

We would also thank Fridtjof Pedersen Lersveen and Benjamin Karlsen for technical support throughout the entire project and Marianne Talberg Andersen for proofreading.

Summary

This project was conducted as a bachelor thesis at NTNU Ålesund, given by Solwr. The main focus of this thesis was to investigate a way to identify, pick and stack paint buckets using a robot and computer vision. The solution aims to simulate the working environment for Grab, Solwr's autonomous warehouse robot. Grab is developed to handle picking tasks [1]. The thesis also presents a research trip to a company handling paint buckets, to observe their routines and how an automated solution could be implemented.

The final solution consists of a Viper 850 robot picking and placing paint buckets, in cohesion with a Lidar 515 camera [2]. The developed result uses software such as Ace [3] and Sysmac [4] to move the robot, and algorithms written in Python to detect the buckets and to place them.

The result is an automated solution capable of detecting, picking and placing paint buckets on a pallet. While the accuracy is not perfect, the bucket can consistently be placed and stacked with some deviation. The error is due to a limitation with the used camera.

Sammendrag

Dette prosjektet er en bacheloroppgave ved NTNU Ålesund, der oppdragsgiver er Solwr. Hovedfokuset ved denne oppgaven var å undersøke måter å identifisere, plukke og stable maligsspann ved bruk av en robot og maskinsyn. Løsningen skal simulere arbeidsforholdene til Grab, Solwr's autonome varehusrobot. Grab er utviklet for å kunne håndtere plukking og plassering [1]. Oppgaven presenterer også et kartleggingsbesøk til en bedrift som er grossist, for å observere deres rutiner og hvordan en autonom løsning kan implementeres.

Den endelige løsningen består av en Viper 850 som plukker og plasserer malingsspann, i samarbeid med et Lidar 515 kamera [2]. Løsningen benytter seg av software teknologi slik som ACE [3] og Sysmac [4] for å bevege roboten, og algoritmer skrevet i Python til å detektere og plassere bøttene.

Resultatet er en autonom løsning som er kapabel til å detektere og plassere malingsspann på en palle. Selv om nøyaktigheten ikke er perfekt, klarer den å plassere bøtter konsistent med små avvik. Feilen forekommer av begrensinger knyttet til kameraet.

Table of Contents

List of Figures

List of Tables

Terminology

Terminology	
Expression	**Description**
Grab	Solwr's automated warehouse logistics robot
Viper robot	6 DOF arm made by Omron
Sysmac	Industrial automation software used for controlling manufacturing equipment
Ace	Industrial automation software used for controlling manufacturing equipment
Point cloud	A representation of 3d space using discrete points
Intel realsense L515	A 3d-camera used to gather point clouds from its view
Gripper	a device that allows a robot to grasp, hold, and manipulate objects
End effector	The device at the end of a robotic arm designed to interact with the environment, performing tasks such as gripping, welding or painting
PLC	Programable-Logic-Controller, industrial controllers used in factories and automation
EtherCAT	Communication protocol used by PLC and similar equipment like robots
TCP/IP	Simple handshake communication protocol over IP networks
COM port	Serial port interface used by computers for input/output, often using the RS-232 standard
OPC UA	Universal standard for sharing variables and information among PLC's works on a server client system
OPC DA	Same as OPC UA but COM-port based and outdated
RANSAC	Detection algorithm that works on iterative comparisons
LIDAR	Laser radar. A way to map points in 3d space using a rangefinding laser that scans its surrounding area
Algorithm	A procedure consisting of repetitive steps to solve a problem
HSE	Health Safety Enviromental, related to workplace health and safety regulations
SKU	Stock Keeping Units, a unique code that identifies each distinct product and service for inventory tracking and sales purposes

Table 1: Terminology

1 Introduction

In this report, we provide a comprehensive overview of the development and implementation of a robotic system designed for the picking and placing of paint buckets. This project aims to address various technical challenges and demonstrate the feasibility of using advanced robotic technologies in warehouse environments.

1.1 Background

The assignment was provided by Solwr, a company located in Ålesund, which offers advanced logistics technology for the trade industry. The company has developed an autonomous robot named Grab, seen in figure 1 below. Grab is designed for pick-and-place tasks within warehouse environments. This robot represents a significant innovation in the field of order selection for Stock Keeping Units (SKU's) in conventional warehouses, requiring minimal to no modifications to the existing infrastructure. It is particularly effective in minimizing operational expenses and addressing health, safety, and environmental HSE concerns associated with heavy, repetitive tasks [1].

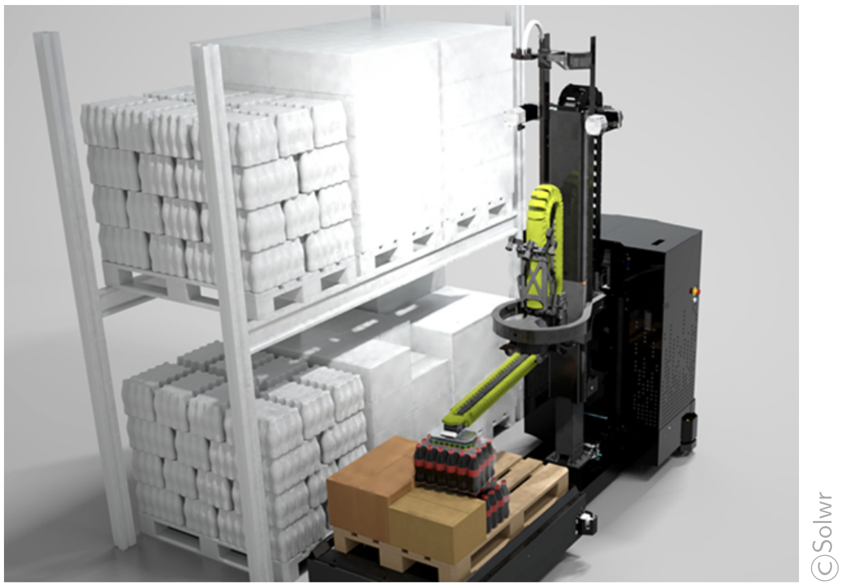


Figure 1: Solwr's Grab [5]

The project's challenge is to use the Grab robot along with a Zivid 2+ camera and computer vision technology, as shown in Figure 2 below, to pick and stack paint buckets.

The challenge is to use Grab along with a Zivid 2+ camera, as shown in figure 2 below, to pick and stack paint buckets. The task involves detecting, grasping, and placing the buckets onto pallets with precision. Despite the apparent simplicity, complexities arise, such as adapting to cylindrical shapes, managing weight effectively, developing reliable picking and stacking mechanisms and overcoming operational hurdles. The objective is to devise a practical solution that addresses these challenges, demonstrating the robot's ability to handle the paint bucket process seamlessly.

Figure 2: Zivid 2+ [6]

1.2 The Process Today

As of today, Solwr primarily handles picking objects up to 30 kg. They have not broached into the area of paint cans, and want to explore the concept. While they do posses the technology to identify and pick cylinder like object, they would like a deeper investigation to better understand the process in a warehouse setting.

1.3 Limitations

Since all local Grab robots are currently under testing or construction, a Viper robot, shown in figure 3 below, will be used for this project. This decision is based on the Viper's constant availability at the Manulab on campus, ensuring access at all times. Viper is an articulated robot capable of performing a range of tasks including machining, assembly, and material handling [7].

Figure 3: Viper 850 [7]

As all Zivid 2+ cameras are tied up to Grab and not easily accessible; an Intel L515 camera, shown in figure 4 below, will be used instead. To mitigate the risk of spills, paint buckets will be empty during the experiments. These substitutions ensure continuous progress and practical testing while addressing the project's logistical constraints.

Figure 4: Intel realsense L515 [8]

1.4 Problem Definition

The task is to utilize and explore potential technologies capable of picking paint buckets, and the potential related challenges. The Viper 850 will be used for picking and placing the paint buckets. The given task requires the Viper robot to be capable of picking the buckets from one spot and placing them on a pallet. This setup simulates Grab's functionality of picking buckets from a storage shelf and placing it on a pallet.

Computer vision will be used in order to locate the buckets, since this is what grab uses today. The camera needs to be capable of detecting the location of paint buckets consistently using point cloud technology.

The placement of the buckets are decided autonomously through an algorithm and are placed with stability in mind where bigger buckets are placed first.

1.5 Manufacturing lab (Manulab)

Manulab at NTNU Ålesund is a manufacturing lab oriented towards improvements and innovation in production technologies as seen below in figure 5.

Figure 5: Manulab at NTNU Ålesund [9]

The facility integrates a diverse set of machinery and technologies across multiple disciplines. These components can function independently or be combined to create fully automated systems. Such configurations are ideal for testing innovative ideas or developing proof-of-concept prototypes, aiding businesses in assessing potential investments in new technologies before making substantial financial commitments [10].

1.6 Research

As part of our research, we conducted a visit to Westing, a supplier of quality-assured paint systems for ships and industry. Our objective was to conduct a comprehensive survey of their current methods for picking and stacking paint buckets, aiming to draw insights as part of our research. We received valuable information regarding their warehouse procedures and regulations concerning the storage of paint/chemicals, as well as HSE regulations.

Westing offers paint cans in various sizes, including 1, 2, 3, 5, 20, and 25 liters, with the 25-liter and 5-liter cans being the most popular. While the buckets usually come in standardized sizes, their weights can vary as some products use the same bucket but are filled with less paint.

Various types of paint, along with items like thinners, soaps, and chemicals, are organized in dedicated areas or shelves within the warehouse. The layout is designed to group similar items together to streamline the picking process. Larger and heavier buckets are stored closer to the order processing area for easier access, while smaller buckets are placed towards the rear of the warehouse. In some instances, a thin layer of cardboard is placed between the first and second layers of buckets to provide stability and prevent vertical stacks from tipping over.

As seen in figure 6, the lowest level in the racks has two pallets stacked on top of each other, while the higher racks have a single layer of pallets. Pallets above 2 meters are wrapped in plastic for safety.

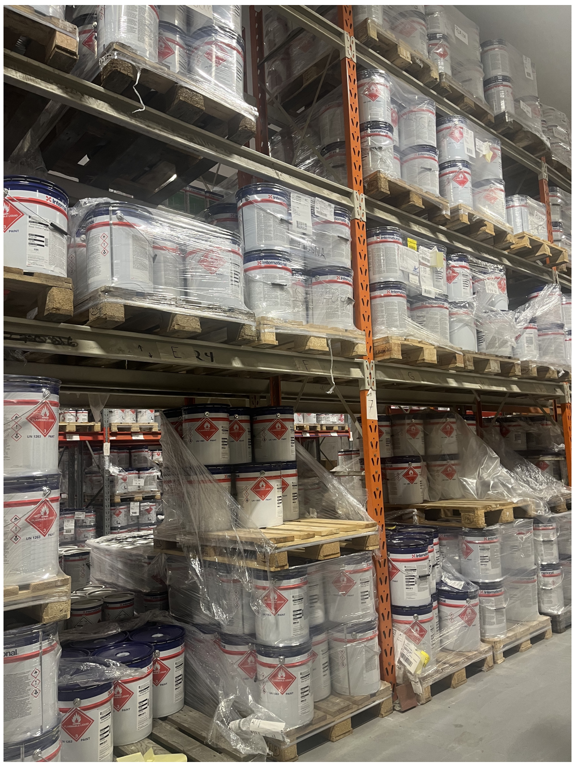
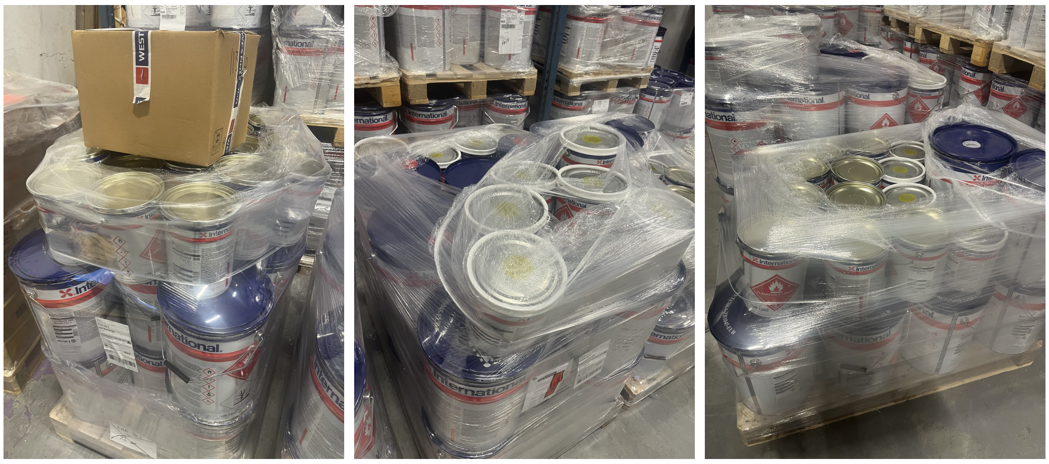


Figure 6: Warehouse rack-layout

Customer orders are organized with the heaviest items at the bottom, forming two layers from the ground up, sometimes with a lighter layer on top. The width of the stack matches that of a euro pallet, and the maximum weight is based on safety standards and the pallet's capacity. The focus is on creating stable stacks without compromising safety. A typical order includes a variety of products and sizes, stacked directly on top of each other without any separators between the layers, as seen in figure 7.



(a) Stack example 1 (b) Stack example 2 (c) Stack example 3

Figure 7: Examples on how they stack their pallets

Orders are manually received and checked in an office before the picking process begins, typically labeled by urgency. Staff return to the office to check for new orders, then pick up a pallet and start stacking. Once an order is prepared, it is wrapped on a rotating platform that also allows for weight checking, marked with an order label, and positioned by the warehouse door for transport. This area also collects empty pallets for order assembly. Finished orders are picked up by a truck and delivered to customers. In urgent cases, Westing handles the deliveries directly.

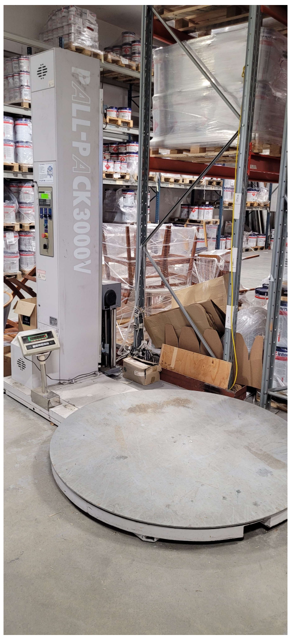


Figure 8: Wrapping area

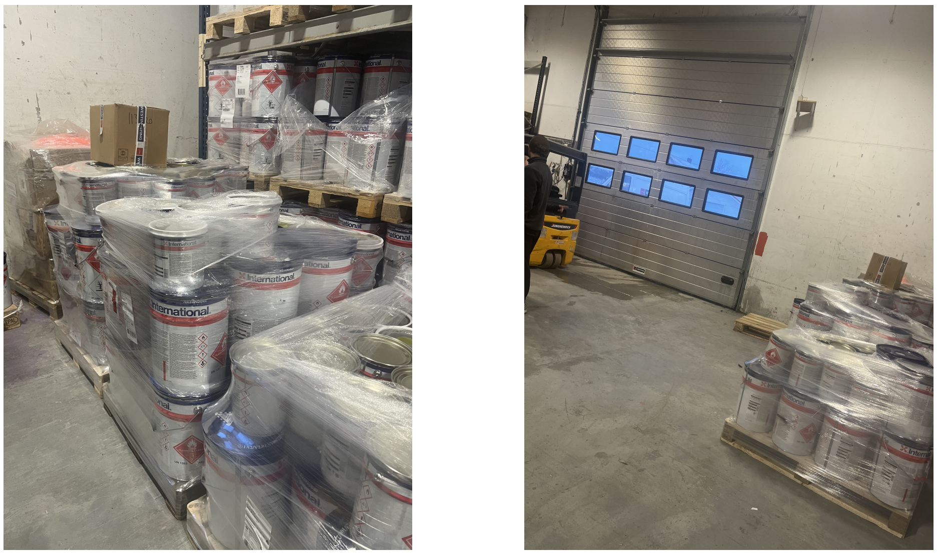

(a) Order/pallet location (b) Pickup location next to orders

Figure 9: Order and pickup location

The number of pallets Westing stocks each day varies by season. On slow days, they handle about 12 orders, which is about to 2-3 pallets. During the summer, this can increase to up to 30 orders a day, though not every pallet is full. Each order includes a mix of different types and sizes of products.

Once a week, a truck delivers approximately 17 pallets of goods to the warehouse. The warehouse is continuously restocked with items they have in stock.

Westing did not encounter specific challenges in picking paint buckets, apart from the heavy lifting involved. Buckets are lifted by hand, but improper lifting can lead to injuries and long-term health issues, resulting in employee sick leave.

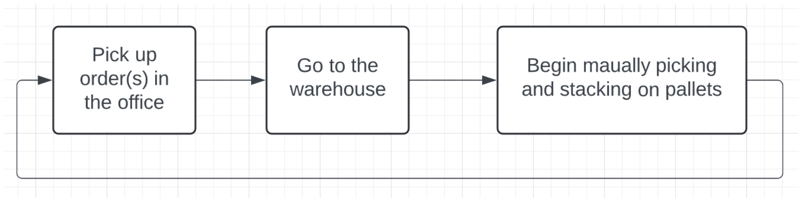

Pick up order(s) in the office
Go to the warehouse
Begin maually picking and stacking on pallets

Figure 10: Flowchart of Westing's picking and stacking process

Westing's primary goal is not to automate the entire process but to eliminate heavy lifting, thereby reducing sick leave.

Westing handles hazardous chemicals on a limited basis since they are sealed in containers. Standard safety measures, including safety shoes, glasses, and gloves, along with HMS safety regulations, are implemented. If paint needs to be emptied due to leakage or deformation, it is either poured into a collector or dried and disposed of as hazardous waste. Customers are unconcerned with minor container defects as long as they are sealed and functional. Chemical exposure incidents follow general procedures involving safety data sheets and safe removal of spills.

If someone comes into contact with a chemical substance, the general procedure is followed: the safety data sheet is taken to the doctor for presentation, chemical spills are removed safely according to the safety data sheet's handling instructions, and the incident must be logged.

2 Theory

2.1 Viper 850

The Viper 850 is an six-axis articulated robot optimized for high-speed and precise automation tasks in industrial settings. The different axis can be seen in figure 11 below. As seen in table 2 , it features an 855 millimeters reach, and is designed to handle tasks requiring both speed and accuracy.

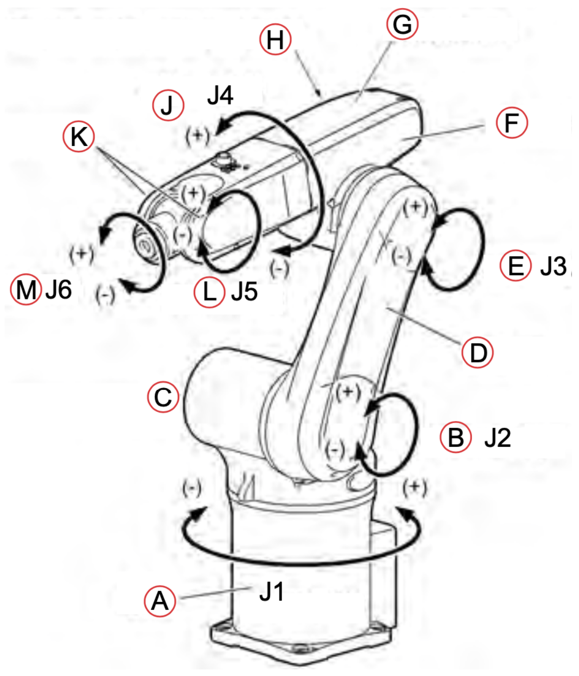


Figure 11: An illustration of the viper axis [11]

Technical specifications	
Reach	855mm
Payload	Up to 5 kg
Repeatability	±0.03 mm
Degrees of freedom	6-axis

Table 2: Technical specifications [12]

The Viper 850 is integrated with Omron's control systems. The controller is designed for high processing speeds, ensuring real-time responsiveness and accuracy in robotic operations.

The robot supports multiple communication protocols, including EtherCAT, Ethernet/IP, ProfiNet and DeviceNET [12].

2.2 Suction Cup

In order to calculate the size and force needed to lift an object using vacuum we can apply the following formula:

$$W = \frac{P \cdot 100{,}000 \cdot A}{9.81} \tag{1}$$

Where W is the weight of the object in kilograms, P is the pressure differential in bar, A is the area of the suction cup in m^2 and g is the acceleration due to gravity in m/s^2.

When using suction cups for lifting objects its critical to ensure proper sealing and proper distribution of contact points relative to the objects centre of mass. Moving containers with liquid inside will cause tank slosh around and shift the centre of mass. Additionally as the liquid oscillates back and forth it will produce a rocking effect that can cause the bucket to dislodge. This will at best make stacking difficult and at worst cause material or personnel harm. Therefore it becomes critical to ensure a proper rigid grip.

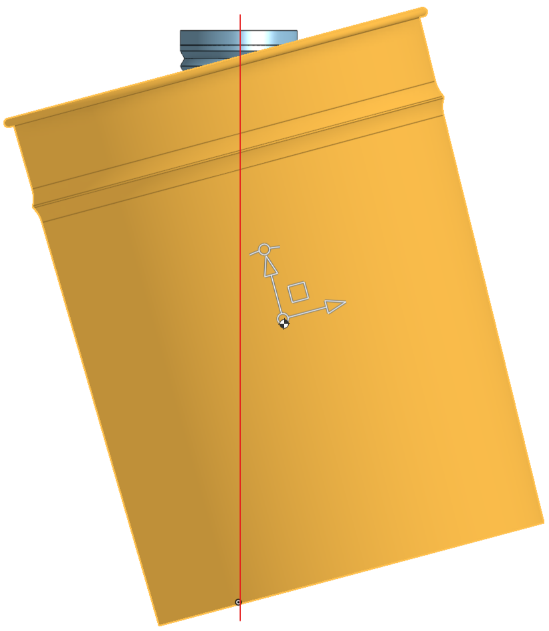


Figure 12: Representation of centre of mass under movement due to inertia and sloshing

One such method could be a compartmentalized suction cup, where the suction cup is divided into smaller sections that can move independently as seen in figure 13. This would allow the suction cup to adapt to the shape of the object and ensure a proper seal. This would also allow the suction cup to adapt to larger or smaller objects depending on the layout and design of the gripper. However this also increases the complexity of the system and the risk of failure as vacuum valves are needed to control the individual sections.

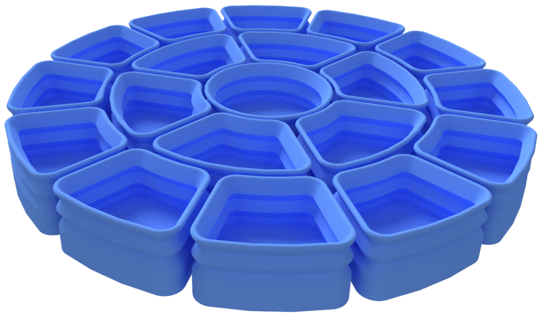


Figure 13: Gripper with compartmentalized suction cup

Another method could be inserting hard contact points inside the vacuum cup as seen in figure 14, as the vacuum pulls the object close the hard contact points will press against the object and ensure a rigid grip. This simplifies the design greatly at the cost of being a single point of failure should the seal break.

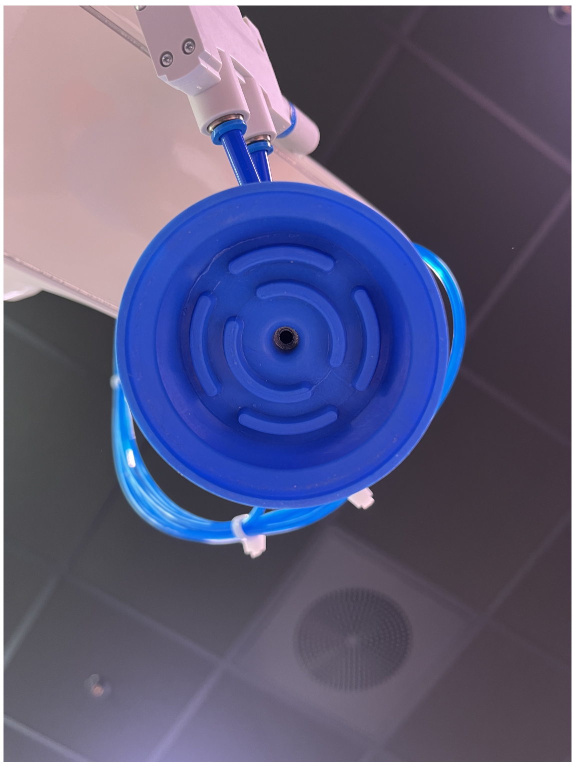


Figure 14: Gripper with hard contactpoints built inside the vacuum chamber

2.3 Sysmac Studio

Sysmac Studio is Omron's automation software designed to support the development, configuration, and maintenance of industrial automation systems. It provides an integrated development environment (IDE) that supports multiple programming languages including ladder, structured text, and function block diagram, following the IEC 61131-3 standard. The software includes advanced simulation tools that enable testing and validation in a virtual environment, reducing development time and errors [4].

Sysmac Studio uses precise motion control and synchronization for complex automation tasks and integrates seamlessly with Omron's NX/NJ series controllers. Its user interface is designed for ease of use, with intuitive navigation and comprehensive documentation. The software also features diagnostic and troubleshooting tools to efficiently identify and resolve system issues [4].

2.4 ACE 4.0 (eV+)

ACE, Omron's automation control environment, is designed for programming and managing robotic systems using the eV+ language.

eV+ is a proprietary, task-oriented language made for robotic control and automation. It offers a robust set of commands and functions that enable precise control over robotic movements, data management and interaction with peripheral devices. This language is specifically designed to meet the demands of real-time robotic applications [13].

The ACE software provides a user-friendly interface to program, control and monitor robots using eV+. It includes various tools for robot calibration, vision system integration and motion planning, among other features, facilitating the deployment and management of robotic applications in industrial settings [3].

2.5 Open Plattform Communications (OPC)

Open Platform Communication (OPC) is a series of standards and specifications for industrial telecommunication. Developed to facilitate interoperability between different hardware and software in automation systems, OPC ensures seamless data exchange [14].

2.5.1 OPC Data Access (OPC DA)

OPC DA focuses on the real-time exchange of data between devices and control systems. It operates over Microsoft COM/DCOM technology, allowing applications to communicate with industrial hardware such as PLCs, DCSs, and SCADA systems. OPC DA is widely used for acquiring and monitoring real-time data, controlling devices, and ensuring timely updates [15].

2.5.2 OPC Unified Architecture (OPC UA)

OPC UA is an evolution of the OPC standards, designed to address the limitations of OPC DA. It provides a platform-independent framework that enhances security, reliability, and interoperability. OPC UA supports complex data models and offers features like data encryption, authentication, and a robust communication protocol that works across various platforms and networks. Unlike OPC DA, OPC UA can run on multiple operating systems and supports both binary and web service communication, making it highly versatile for modern industrial applications [16].

2.6 Python

Python is a programming language designed to support multiple programming modules. Python features a broad area of usability, where the language is created to handle multiple programming paradigms, examples of these are functional programming and object oriented programming. Python is a high level language designed around readability, including a broad standard library. Python also features a wide selection of third party modules created by the user community. The program has vast documentation available and has a versatile application, making it convenient to use in developing algorithms [17].

2.6.1 SQLite3

SQLite3 is a module in the standard Python library that allows users to create and edit databases, while creating a cross-platform database file. The module is self-contained, so there are no requirements for a system or server to be able to operate within the database. A cursor is used to manipulate tables and information in the database. After creating a complete table, a database file will be created upon running the script once [18].

2.6.2 Matplotlib

Matplotlib is a library in Python that allows for creations of plots and other visualisations. These visualisations can be interactive, and is highly customizable [19].

2.6.3 Numpy

The Numpy library in Python is used to handle mathematical operations involving arrays. It provides useful tools for calculating mathematical operations, including linear algebra [20].

2.7 RANSAC

RANSAC stands for ”Random sample consensus” and is an algorithm that estimates a data model to an existing dataset. It works by grabbing enough data points to extrapolate an mathematical model, like a polygon, circle, line or other rule set; then comparing the dataset to see how well it overlaps with the initial data points

[21]. For example, if you are to find a straight line in a noisy dataset you would select two points at random, draw a line and check the distance to all the points. then you check how many points are within tolerance and try again until you find the result with the highest number or a result that's good enough. The advantage of RANSAC is that its less prone to clustering of outlier data, where a mean function would be weighted by outliers a RANSAC algorithm can look past them as long as they don't behave exactly as the model you are looking for [22].

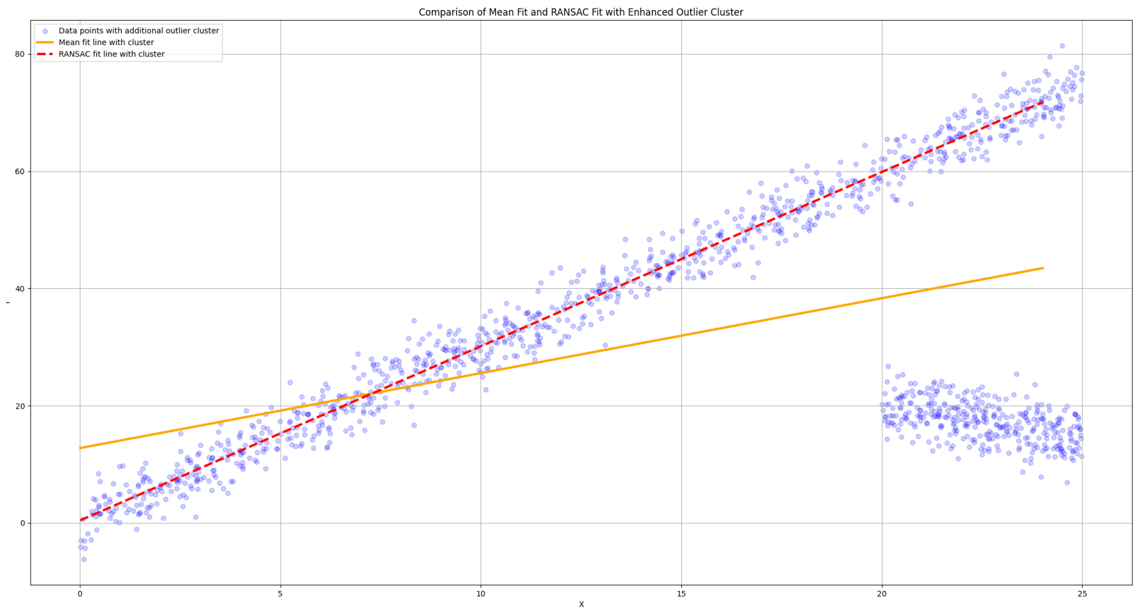


Figure 15: Illustration of how ransac searches for a line compared to a mean function

2.8 Bilateral Filtering

Bilateral filtering is a method of smoothing out data. The method applies a filter (often Gaussian) on nearby datavalues, but preserves sharp transitions like an image with black and white border. Thus allowing similar regions to together without bleeding over from nearby outliers [23].

the bilateral filter is defined by the following equation:

$$\mathcal{N}_r(p) = \{q \in \mathcal{P} | \|q - p\| \leq r\} \tag{2}$$

This equation defines the neighborhood of a pixel p as all pixels q within a radius r of p.

$$\delta p = \frac{\Sigma_{q \in \mathcal{N}r(p)} w_d(\|q - p\|) w_n(|\langle \mathbf{n}_p, q - p\rangle|)\langle \mathbf{n}_p, q - p\rangle}{\Sigma_{q \in \mathcal{N}r(p)} w_d(\|q - p\|) w_n(|\langle \mathbf{n}_p, q - p\rangle|)} \tag{3}$$

- δp is the filtered pixel value
- $w_d(\|q - p\|)$ is the spatial weight function
- $w_n(|\langle \mathbf{n}_p, q - p\rangle|)$ is the range weight function

- $\mathbf{n}_p$ is the normal vector at pixel p
- $\langle \mathbf{n}_p, q - p \rangle$ is the dot product of the normal vector and the vector between p and q
- $\mathcal{N}r(p)$ is the neighborhood of pixel p

2.9 Off and On-Line Algorithm

Off-line algorithm operates under the assumption that the entire sequence of data is available before any processing begins. This way a script can utilize the best optimization of allocations [24]. This method is suitable for handling orders, where the content in a sequence is known before being processed.

An alternative to off-line algorithms are on-line algorithms. An on-line algorithm is based around finding the best placement for an item, without knowing what the following item in the sequence will be. Such a solution is unsuitable for problems handling pre-determined sequences where best cost is the objective [24].

2.10 Best Fit

A best fit approach is based around finding the best placement for each placement in a sequence. By finding the smallest area available for housing placements, an optimization of available area is achieved. This method is applied when trying to avoid wasting space when placing inside a container with limited space [25].

The problem can be simplified if a specific requirement is present. One such requirement can be a sorting criteria by size. When sorting by size, the order of items can be pre-sorted. The problem is then a matter of finding a suitable location for each item concurrently.

3 Materials

3.1 Hardware

System	Component	Type	Qty.
Robot cell	Viper	Omron Viper 850	1
	PLC	Omron NX102-1120	1
	Robot controller	eMotionBlox-60R	1
Vison	Vision camera	Intel RealSense L515	1

Table 3: Materials

3.2 Software

The following is a table of all softwares used in this project:

Software	Used for
ACE	Programming the robotic arm
Sysmac studio	Programming the system PLC
Python	Point cloud, coordinates for picking/stacking
OPC Expert	Bridge software that converts OPC DA to a UA server

Table 4: Software table

Python libraries	Description
numpy	Math functions library
matplotlib	Graph plotting library
sys	System specific functions
random	Random number generator
os	Operating system specific functions
open3d	Visualization and filtering
pyrealsense2	SDK for lidar camera
math	Math functions library
sqlite	SQL database library
opcua	Communications library for OPC UA
time	Timekeeping library
mpl_toolkits.mplot3d	Addon for matplotlib

Table 5: Python library table

3.3 End Effector

The following inputs and outputs were used for the end effector:

I/O signals		**Used for**
Output	6	open (turns off the suction)
Output	5	close (turns on the suction)
Output	0	not used
Input	0	not used
Input	0	not used

Table 6: End Effector - I/O signals

4 Method

This section explains the practical navigation of the thesis and the choices that were made. This includes the work process planned and performed by the group. For the equipment used by the robotic arm; the suction cup is described. Following this, the software used and developed is listed detailing the process. These include algorithms written in Python, and programs written using Ace and Sysmac. Finally, the algorithm for locating paint buckets is explained.

4.1 Work Process

In this project, we employed a SCRUM approach. This method was particularly effective in the early stages when several changes were required. SCRUM enabled us to adapt quickly to these modifications. This approach allowed for continuous refinement and improvement of the project.

Each team-member would be assigned tasks as "lead" and also have at least one other team-member as "helper". These tasks were organized such that the member helping would have to cooperate later to integrate each component together. To define each group member's responsibilities and keep deliverables on track, we made a Gantt chart that can be seen in figure 16 below.

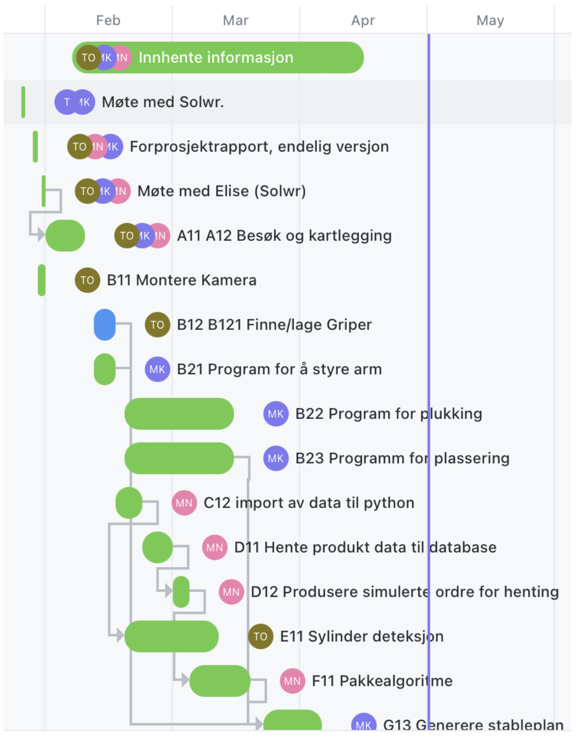


Figure 16: Gantt chart

In order to develop a solution, we had several objectives that needed to be looked into. Some key points were identified:

1. **Assess requirements:** The first objective was to conduct a visit to Westing, and learn about their warehouse procedures and regulations. During our visit we gathered information about typical routines and identified potential hurdles. The work done during this visit would lay the groundwork for future decisions.

2. **Testing hardware:** Movement testing with the robot arm was done to see how the Viper performed. As it was deemed suitable for this objective, it was necessary to choose a fitting suction cup for picking the paint buckets. The suction cup had to be able to pick several sizes of paint buckets.

3. **Establishing communication:** It was important to establish communication between the different softwares. Thorough research had to be done to find the best fit for this project.

4. **Detection:** Manulab provided a Intel realsense L515 camera for the duration of the project. With it, a program capable of identifying the paint buckets was created.

5. **Acquire test material:** To do the pick and place testing, it was necessary to obtain paint buckets. These buckets had to be of varying sizes, and had to be typical buckets that would be found in a warehouse. A testing pallet was developed specifically for the Viper cell in Manulab.

6. **Stacking:** A program being able to handle the stacking of multitude of paint buckets was developed in Python. This included creating a database containing the paint buckets that were used for testing.

7. **Integration:** Slowly as more and more parts were finished, all parts got integrated into one cohesive program and holistic testing could start.

4.2 System Overview

Our system architecture can best be illustrated by looking at figure 17 below.

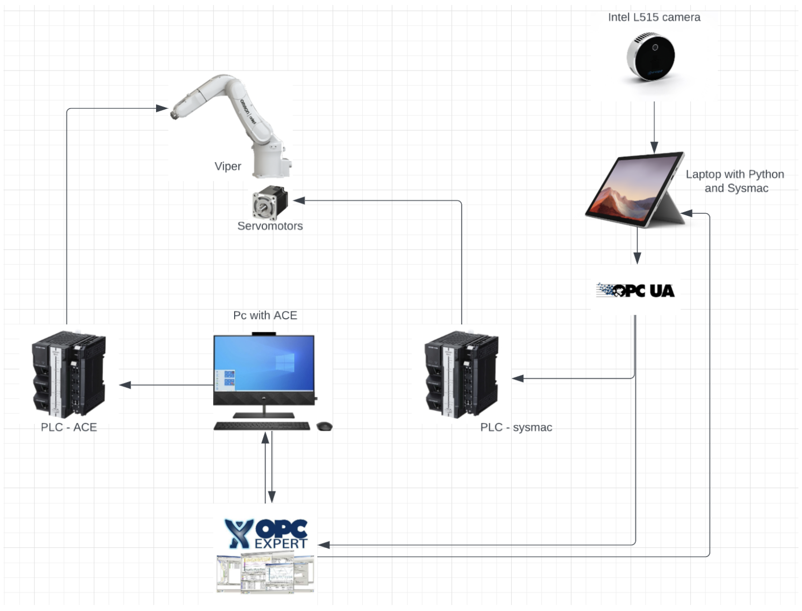


Figure 17: This image gives a quick overview of how the implemented system works

4.2.1 Order of Operations

To better explain how data flows in our system, we can describe it by the following operation. This it not a step by step operation, but rather an overall description to help understand how the system works from input to movement.

1. There is one laptop that works as a master and relays information to all other entities. It takes in continuous frames from the lidar camera and runs the picking interface. Since the viper arm and linear axis can not be controlled by a singular PLC the data flow splits into two lanes.

2. The first connection bridges directly between python on the laptop straight to the PLC controlling the linear axis.

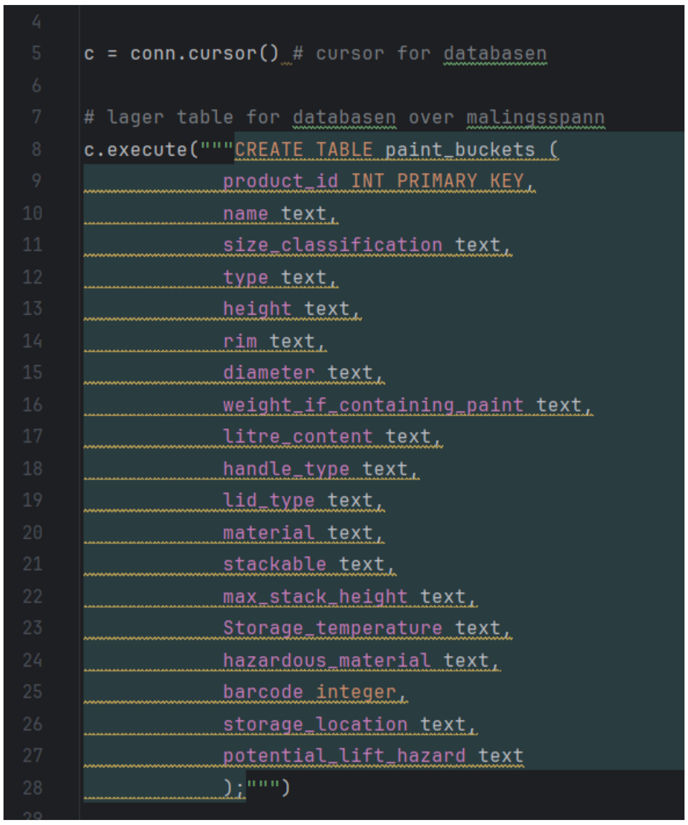
3. The second connection is a bit more complicated as it cannot talk directly to the vipers PLC due to incompatible OPC generations.
Instead OPC commands from python gets converted through the middle ware program that feeds into ACE running on a windows machine inside the viper cell. ACE then acts on incoming signals and moves the arm accordingly.

4. Through these lanes, commands and check flags are sent back and forth to ensure proper timing.

4.3 Python

4.3.1 Paint Bucket Database

The database was created using SQLite3 in Python. Firstly, the script creates a connection to the database. A cursor is created to allow the script to manipulate data. A table was created to store information about each paint bucket.



```python

c = conn.cursor() # cursor for databasen

# lager table for databasen over malingsspann
c.execute("""CREATE TABLE paint_buckets (
            product_id INT PRIMARY KEY,
            name text,
            size_classification text,
            type text,
            height text,
            rim text,
            diameter text,
            weight_if_containing_paint text,
            litre_content text,
            handle_type text,
            lid_type text,
            material text,
            stackable text,
            max_stack_height text,
            Storage_temperature text,
            hazardous_material text,
            barcode integer,
            storage_location text,
            potential_lift_hazard text
            );""")
```

Figure 18: Snippet of the paint bucket database script

In the database information on a selection of empty buckets that were obtained for testing is stored. The paint buckets obtained were of varying sizes, which helped simulate an actual warehouse order during testing. There was a need to create detailed descriptions of each paint bucket, including necessary information such as height and radius, shown in figure 18.

After completing the database, a function called get_information_by_id was created where upon entering an existing ID from the database would print out information about the specific bucket. This function was later integrated into the main script containing the packing algorithm for a more convenient user experience. The content of the completed database is viewable in SQLite viewer, allowing for easy sharing and viewing of the file online [18].

The database contains a variation of paint bucket sizes. The buckets were grouped into four main size categories with small variations within each size group. This was later used by the packing algorithm to help visualize the different buckets trough colors for each size group.

Upon visiting Westing, it was stated that an automated process taking care of heavy lifting would reduce the amount of sick leave amongst employees. This was taken into consideration when creating the database, as all paint buckets over 15 kg is labeled as a potential lifting hazard. The potential weight of every paint bucket was then included if they were filled, based on an estimated weight for each bucket.

The script features several fields to store data in a table. The fields are as listed:

- product_id, an INT PRIMARY KEY containing a unique ID for each paint bucket in the database. This ID is the numerical value entered by the user to extract information or input orders. These numbers correspond with the numbers marked on the physical buckets used in testing.
- name, a text record containing a unique descriptive name for each bucket. Some buckets share the same features, so they will also share the same name description. This field is to help identify the buckets separately from their given numerical value.
- size_classification, a text field where the paint buckets were sorted into a group of similar sizing. The database contains 4 sizes, xlarge, large, medium and small. Mainly used for color visualization between sizes in the plot. Some size variations are shown in figure 19.
- type, a text field describing what type content the buckets have. Each bucket in the database is listed as a paint bucket in this objective.
- height, a text field containing the height of each individual paint bucket corresponding to the measurements taken of the physical buckets.
- rim, this field contains information about the rim measurement of each bucket, which is used to help calculate the placements.
- diameter, a text field containing the diameter of each individual paint bucket corresponding to the measurements taken of the physical buckets, also used to calculate placements.
- weight_if_containing_paint, this text field contains a rough estimation of the weight of each bucket would be if it was filled with maritime paint. The paint buckets used for testing in this project were all empty due to the lifting limits of the viper robot arm, and due to the risk of any content spilling in the Manulab testing environment.
- litre_content, this text field contains an estimated calculation of the litre content in each paint bucket if they were filled with paint.
- handle_type, a text field where the handle type on each bucket is listed. These are listed as there was an interest in mapping if any of the handles on the buckets would restrict Grab's ability to pick them. None of the paint buckets

used for testing in this project is assumed to feature any handles that could complicate lifting.

- lid_type, this text field contains information on all the lid types. The lids are either listed as metal or plastic. This field was included if there was a possibility of any lid having features that would complicate lifting with the suction gripper. None of the paint buckets in the database are assumed to have any complicating features.

- material, a text field containing information about what material the paint buckets are made of. In this database the buckets are either listed as metal or plastic.

- stackable, a text field containing information if any of the paint had any features that would make them challenging to stack. None of the paint buckets in this database had any features that would complicate stacking.

- max_stack_height, this text field is just included as a given height of 2 metres, as pallets above 2 metres are wrapped in plastic. This This is done as a safety precaution.

- storage temperature, this text field contains the information about the minimum storage temperature the paint buckets should be stored at. The storage temperature should be above 5 degrees.

- hazardous_material, this field contains information about whether the paint content in the paint buckets are hazardous. All he paint buckets in the database are labeled with yes, as maritime paint can be hazardous to humans and the surroundings.

- barcode, this integer contains a unique barcode for each paint bucket.

- storage_location, the location the paint buckets were stored at. The location was listed as Manulab for this objective.

- potential_lift_hazard, this field will inform about potential lift hazard when manually lifting the paint buckets. All buckets above 15 kg is labeled as a potential lift hazard. This limit is set due to the strain lifting up to and over 15 kg can put on the human body over time.

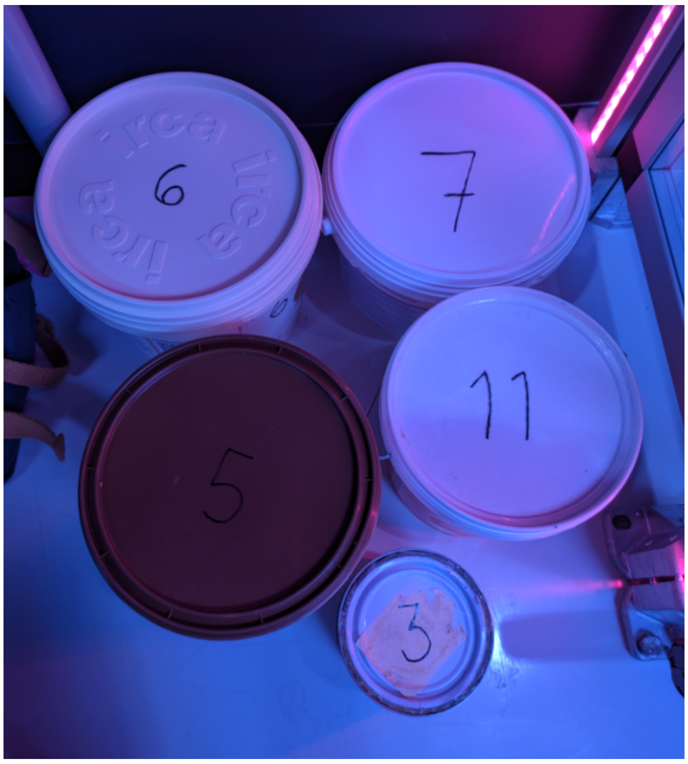


Figure 19: A selection of the buckets saved in the database

4.3.2 Packing Algorithm Script

The script containing the packing algorithm was written using Python. AI was used to assist during the development and debugging of the algorithm, to help improve the quality of the functions that were ultimately used.

The following is a list of the modules used:

- SQLite3 allows the code to create a connection to the SQLite database and extract information from it to be handled in the script. By using this module it is possible to extract and manipulate data from the paint bucket database.
- The numpy module is used to calculate dimensions and positions for the paint buckets based on the data extracted from the database. Generated arrays will define spatial coordinates for the paint buckets, represented as cylinders in the 3D plot.
- Matplotlib, this library allows the script to generate 3D plots for the stacking, allowing for a visual inspection of the placements by the algorithm. From this library the modules mpl_toolkits.mplot3d is imported to help render a visual representation of the cylinders. In the plot, the sizes of the paint buckets are represented by measurements that were obtained from a varied selection. Matplotlib also takes care of customization for viewing angles of the plot, where the different sizes of buckets are represented by respective colors to make the visual inspection easier.

The first iterations of the script focused on picking the largest buckets first for stacking, and then moving on to stack smaller sizes. This way the heaviest buckets would be placed first on the pallet to ensure more overall stability. The algorithm was developed by researching packing algorithms, focusing on placing circles inside a square [26].

The script uses an offline approach, attempting to find the optimal fit of each paint bucket in a given area. The offline approach was chosen as the testing simulation done emulates a warehouse scenario. The content in a warehouse will be known and listed before the buckets are picked and placed [24].

Matplotlib was used to create a visual plot of the ensuing stacking. Matplotlib was chosen as it is the most popular and intuitive Python library for visually representing data. The module contains all the tools needed to create a visualisation of the orders handled by the script [19].

The script is written to utilize the space of any pallet, where the measurements can be changed as needed in the script. The working principle in this script is based around sorting the paint buckets by size and then trying to find the most efficient placement for the paint buckets. A few rules was implemented for the script, trying to minimize the wasted space and trying to ensure a stable stacking:

- By sorting trough and processing the buckets by size, the heaviest paint bucket size is chosen and stacked first, only moving on to smaller sizes when all larger sized buckets are placed on the pallet. This was done by sorting every bucket in an order by utilizing the diameter and height field of the buckets ID's in the database.
- When stacking, the script checks for available space for each placement at the lowest available space. It must find a placement with no overlap of already placed buckets, and placement without exceeding the pallet boundaries. Only smaller buckets or buckets of similar sizing can be placed on top of another bucket.
- The initial height map is set to zero. Upon successfully placing a bucket, the height map on the pallet will be updated. The height map represents the current height of every position on the pallet, managing the vertical stacking. It was decided that a maximum of two paint buckets were allowed to be stacked vertically during testing in Manulab.
- It was also later on implemented that the stacking had to be done with a distance of 1 cm between each paint bucket. This way it could be ensured that smaller buckets stacked on top of another bucket would be placed more centered, avoiding unstable placements. This is due to the bucket being placed a total of 1 cm away from the bucket underneaths rim.

The first iterations of the algorithm focused on picking the largest buckets first and then moving on to smaller ones. This way it was ensured that the heaviest buckets would be placed first on the pallet to ensure more stability.

It was important to get a visual plot of how each stacked stacked order would look. In the script it was opted for 3 different overview plots of an order, where the angles could be adjusted as needed. The first plot showed the stacking from a direct horizontal angle, the second plot showed an angled view where the elevation was set to 60 degrees and the diagonal angle set to 45 degrees. The final plot showed the stacking directly from above. The plots feature cylindrical shapes representing each bucket in any order, where the color will represent their size classification. In figure 20 an earlier plot is showcasing the stacking.

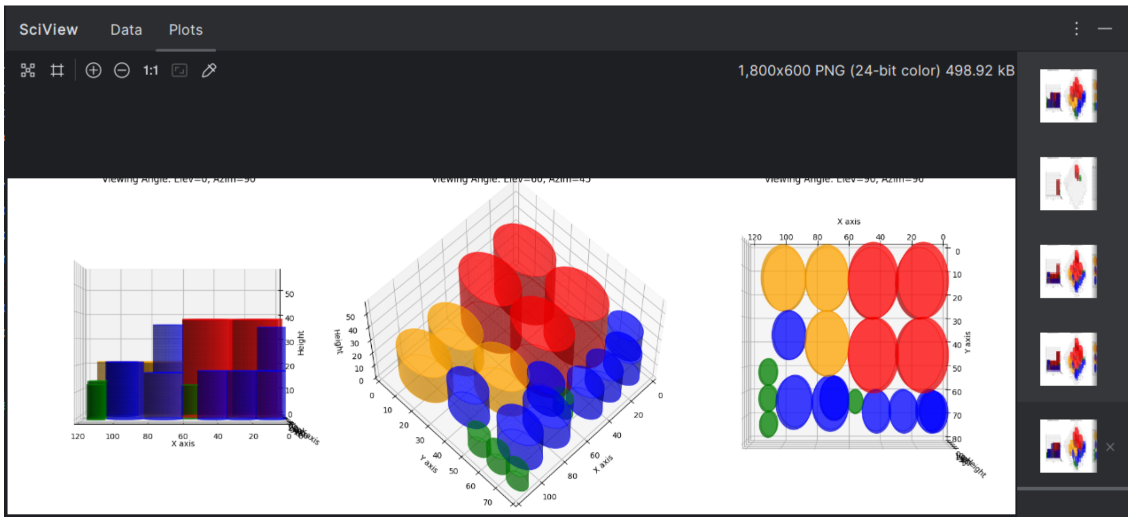


Figure 20: A plot of an order containing 19 paint buckets from an earlier version of the script

The algorithm allows for several orders to be entered by the user at once. Once orders are entered a 3D plot of the resulting placements are visualized via Matplotlib. The plot is a vital feature for understanding the stacking, and is part of creating a clear and better user experience. Later on it was implemented an option to confirm if the plots were of a satisfactory layout before the picking would ensue. This way it would be possible to redo the order if any mistakes or needed changes were observed in the plot.

Later iterations of the code focused on handling different input choices, integrating the get_information_by_id function. Typing in different keys will navigate the user trough the menu.

The placement logic needed to be improved for the paint buckets stacked on the second layer. The placements needed to be more centered on the buckets placed underneath, to help reduce the instability of the placements. this was done by implementing the rule to have a set spacing between each paint bucket rim to a total of 1 cm, as shown in figure 21. The set distance would ensure that smaller paint buckets placed on top of other buckets would be placed 1 cm within the rim of the bucket underneath. This placement rule also helped avoid disruption by the buckets being placed to close on the first layer as well. With too tight of a fit, the chances of buckets accidentally bumping into contact with each other was deemed too high.

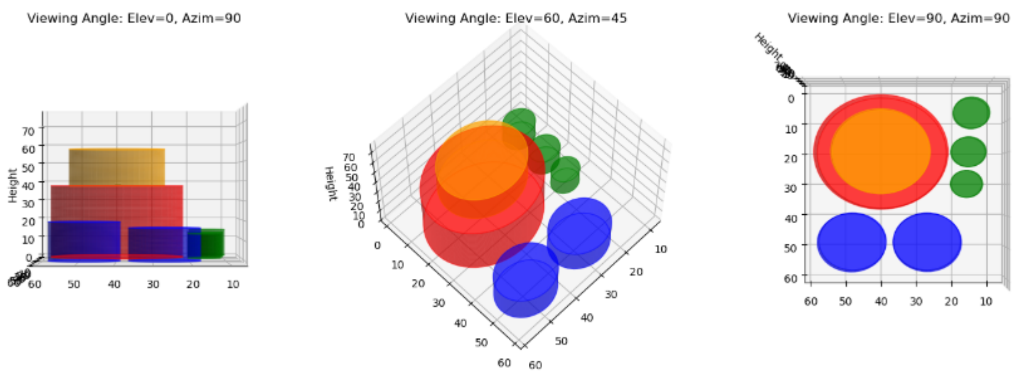


Figure 21: A plot from a later version of the script, showcasing the spacing between the buckets

In the script added to the zip file accompanying this thesis, several functions are written to handle the packing and stacking of the paint buckets.

get_bucket_details
In the first function get_bucket_details, the code will check for the IDs of each paint bucket given as an input. It connects to the product_id in the database file. Trough querying the database for the entered IDs, the IDs will be confirmed to exists or not. If not, the user is alerted to an invalid paint bucket ID.

Upon confirming an existing ID for a paint bucket, information about a confirmed ID is extracted to be stored in the bucket_details list as tuples. The IDs of non-existing buckets will be stored in the not_found_ids list. Entering a valid ID will then result in the information about the paint bucket being printed out to read.

Upon an invalid ID being found in the not_found_ids list, the user is alerted that the IDs were not found in the database trough a printed warning. Entering "done" as an input will complete process and return to the main menu. The function then returns the bucket_details list.

get_information_by_id
In this function the user can query information about any of the paint buckets from the database, given it's ID as input. To query about any bucket ID, the ID must be entered individually, only one ID can be quarried at a time. Entering a valid ID will result in a SQL query being performed to retrieve information from the database. Information about the paint bucket will then be printed as an output to the user, giving a overview of all the information logged about the bucket.

It will alert the user if an invalid input is given, such as a non-numerical value. In that case, it will ask for a valid integer to be entered. It will also print a message alerting the user if an integer not registered in the database is given as input.

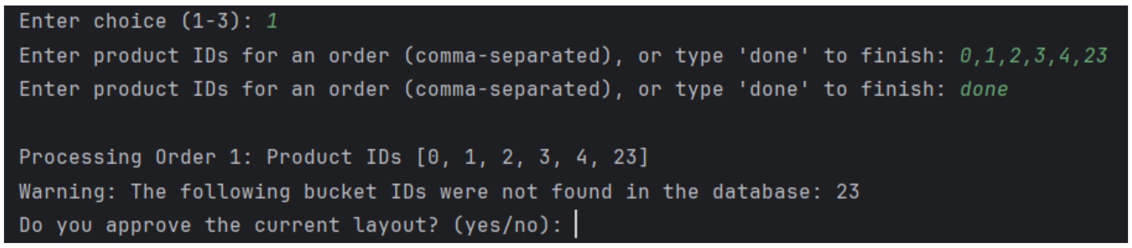
Upon entering "exit" as the input the function will terminate and return back to the main menu.

get_user_input
The function get_user_input is part of the menu presented upon running the script. It handles option "1. Enter orders".

The user will be asked to enter one or several product ID's, keeping them separated by commas. Upon entering ID's, the function will ignore any non-integer inputs and errors such as a empty input. Entering an invalid integer will result in a warning, as shown in figure 22. All valid integers will be added to a list representing the content in an order.

Typing 'done' breaks the loop within the function, and any given orders will be returned. This will then start the process of generating a plot.

Typing 'abort' at any point, will abort the process and take the user back to the main menu, and no plot will be generated if any were given. It gives the user the option to completely exit option 1.



```
Enter choice (1-3): 1
Enter product IDs for an order (comma-separated), or type 'done' to finish: 0,1,2,3,4,23
Enter product IDs for an order (comma-separated), or type 'done' to finish: done

Processing Order 1: Product IDs [0, 1, 2, 3, 4, 23]
Warning: The following bucket IDs were not found in the database: 23
Do you approve the current layout? (yes/no):
```

Figure 22: In this case 23 is not a valid ID

get_confirmation_user
Upon entering an order, the script will generate a plot using Matplotlib. This will then be presented to the user with a request for approval for the layout displayed. This approval is done before the actual picking is instigated by the robot. Upon entering "yes" the picking will ensue, upon entering "no" it will print out a message about the layout not being approved.

Upon entering no, the function prompt_for_order_reentry is called, and an option offering to re-enter the order or to skip it is presented. If the skip option is chosen the script will return to the main menu.

prompt_for_order_reentry
As mentioned in the get_confirmation_user, this function is called in the instance of an order layout not being approved. This function presents the option to re-enter any order or skip it. When choosing to skip the re-entry of an order the function will return false, resulting in the main menu being displayed again.

This function is only called upon a plot not being approved by the user. Re-entering an order will not cause the entire list of orders to be re-entered, as only the specific order will be changed and the script will continue processing the rest of the orders that remain unchanged.

plot_cylinder
In the plot_cylinder function the drawing of the cylinders is handled, using Matplotlib to create a 3D plot.

The function will use calculations to create a cylindrical shape representing the paint buckets. These are hollow shapes that are missing the top and bottom similar to a straw. Since it's desirable to illustrate a complete cylinder a bottom and top will be added to close it. This is done for each unique ID being processed in an order.

To plot the cylinder, it takes the center coordinate of the cylinders base, it's height, diameter and it's vertical starting position which is the z coordinate, to create the shape. By utilizing 50 data points, it will perform trigonometric calculations for each point to find the coordinates to create the shapes. The circles are created using the circle function in Matplotlib.

The cylinders are positioned vertically from the z_start coordinate, which is default at 0 for initial layers, starting the cylinders placements at the bottom of the z-axis in the plot.

visualize_stacking
This function handles much of the visualization done when creating the plot of the stacked orders. The stacking is done on a testing version of a pallet, simulating the testing setup available in Manulab. The pallet used for testing measures 80 x 60 cm.

The cylinders on the pallet are a visual representation of the paint buckets from the database. The cylinders will have different colors depending on their size grouping. Here is a listing of all the size types and their respective colors:

Red = xlarge
orange = large
Blue = medium
green = small
grey = any unspecified sizes

Within these size types there exist some variations between the buckets. Using the graphical plotting functions from Matplotlib, the diameter and height variation of each paint bucket is represented in the plot, similar to their real world counterparts.

can_place_bucket
In this function, it will check if a paint bucket can be placed on a specific location on the pallet or on top of another paint bucket. The function will ensure that no overlap happens while a level stacking is maintained.

When choosing a placement for a paint bucket on top of other buckets, it will choose to place it on a location feasible, where a flat surface is available. It will also ensure that there is clearance between the paint buckets, set to 0.5 cm between each paint bucket. This ensures a total clearance of 1 cm between buckets.

This way it will always attempt to do the most stable stacking for every level. It ensures that the placements happen within the boundaries of the pallet, returning false if any of the placements trespass these boundaries. It then checks the height map, ensuring that the area covering a potential placement is level, at various current top heights.

If the area isn't level, it will return a false. Upon none of the checks failing, the placement of an arbitrary bucket will ensue. This is done for each paint bucket in a order.

If a paint bucket or buckets cannot be placed, it will alert the user with a printed message containing the details of the buckets.

update_height_map
The purpose of this function is to update the height map after a paint bucket has been placed on the pallet. In the function there is a 2D array named height_map representing the surface tops of the paint buckets. In the array there will be an entry for a corresponding point on the pallet, storing the current height of the tallest object at that point.

This way there will be an accurate record of the current height status on the surface of the pallet when the map updates. This calculated new_height is added to the current_height. This helps the sequential placing of paint buckets, avoiding overlap and instability. The function will be called immediately after successfully placing a paint bucket.

packing_algorithm
In this function the placement of a paint bucket order is determined. The function will determine the optimal placement based on the buckets measurements, keeping the placements constrained inside the borders of the pallet.

Looking into the packing_algorithm, buckets are first sorted in order ranging from largest to smallest. This is done by taking the diameter and height for each bucket, sorting them in descending order in a list. This is performed by the sorting function in python.

The function then creates a height map defined as a 2D list. This is a two-dimensional coordinate system which tracks the height coordinate at any given point on the pallet. Since no buckets are placed at the start of the algorithm script, the initial value for the height map is 0.

The algorithm then iterates trough the sorted bucket list, attempting to place each bucket it contains. For each bucket, the algorithm moves along the pallets dimensions using the defined coordinate system. For each coordinate, the algorithm checks if a placement is viable by calling the can_place_bucket function.

The can_place_bucket function firstly checks if a bucket can be placed on the pallet without exceeding it's borders. It then checks if any other bucket has been placed by using the height_map variable. If both conditions are met, where it can be placed inside the borders and no other bucket is occupying the space, it returns true.

Once a suitable location has been found for the bucket, the location is stored and the height map is updated accounting for this new placement. The bucket is then marked as placed, and the algorithm moves on to the next bucket in line.

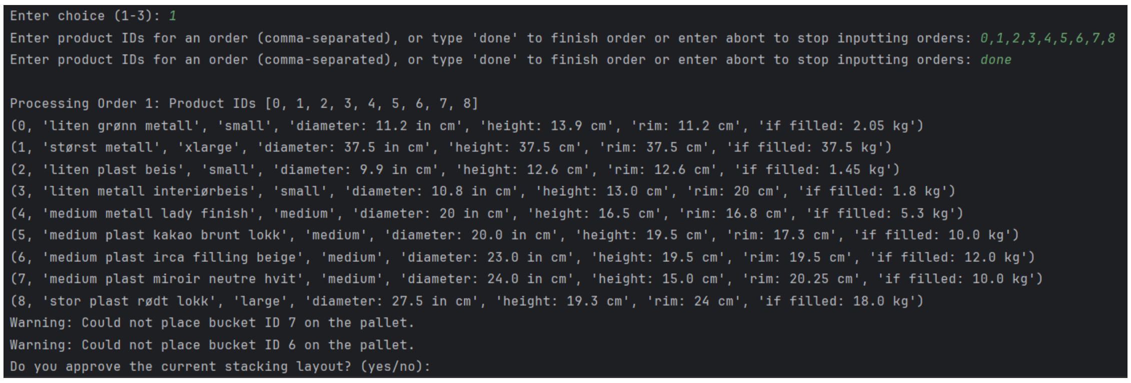
Utilizing this method, the largest buckets will be placed first so that smaller buckets can fill up remaining space. This is done in order to try and achieve stability in the stacking if it consists of a variations of all bucket sizes. The space available for stacking is defined as the pallet_length and pallet_width. In this script the measurements are set to 80 x 60 cm. This simulates the pallet size used for testing in Manulab.

It will also check that the placements won't cause any overlap. This check is performed by the can_place_bucket function, which is called by the packing_algorithm.

simulate_robot_actions
This function takes care of simulating the actions done by the robot to help visualise stacking. For the planned placements the function will do checks for each bucket, checking the locations in the coordinate system and the height placement.

Upon confirming the placements the function will print out how many buckets of each type that have been placed and their coordinates, whether the paint bucket is stacked on top of another bucket. It will then conclude with an printed message informing that the order is complete and ready for transport. Overall it works as a step in the script where placements done by the packing_algorithm are confirmed.

process_orders
Here the script will handle the input of orders. If given multiple orders, the function will iterate trough the list of orders. This will then result in a processing message being printed, instigating the creation of the plot for the first order. I any buckets cannot be placed, the script will alert the user trough a printed message with the details of the buckets not placed as shown in figure 23. The creations of the order plots are done trough calling the visualize_stacking function, displaying an option to approve each plot created individually.



```
Enter choice (1-3): 1
Enter product IDs for an order (comma-separated), or type 'done' to finish order or enter abort to stop inputting orders: 0,1,2,3,4,5,6,7,8
Enter product IDs for an order (comma-separated), or type 'done' to finish order or enter abort to stop inputting orders: done

Processing Order 1: Product IDs [0, 1, 2, 3, 4, 5, 6, 7, 8]
(0, 'liten grønn metall', 'small', 'diameter: 11.2 in cm', 'height: 13.9 cm', 'rim: 11.2 cm', 'if filled: 2.05 kg')
(1, 'størst metall', 'xlarge', 'diameter: 37.5 in cm', 'height: 37.5 cm', 'rim: 37.5 cm', 'if filled: 37.5 kg')
(2, 'liten plast beis', 'small', 'diameter: 9.9 in cm', 'height: 12.6 cm', 'rim: 12.6 cm', 'if filled: 1.45 kg')
(3, 'liten metall interiørbeis', 'small', 'diameter: 10.8 in cm', 'height: 13.0 cm', 'rim: 20 cm', 'if filled: 1.8 kg')
(4, 'medium metall lady finish', 'medium', 'diameter: 20 in cm', 'height: 16.5 cm', 'rim: 16.8 cm', 'if filled: 5.3 kg')
(5, 'medium plast kakao brunt lokk', 'medium', 'diameter: 20.0 in cm', 'height: 19.5 cm', 'rim: 17.3 cm', 'if filled: 10.0 kg')
(6, 'medium plast irca filling beige', 'medium', 'diameter: 23.0 in cm', 'height: 19.5 cm', 'rim: 19.5 cm', 'if filled: 12.0 kg')
(7, 'medium plast miroir neutre hvit', 'medium', 'diameter: 24.0 in cm', 'height: 15.0 cm', 'rim: 20.25 cm', 'if filled: 10.0 kg')
(8, 'stor plast rødt lokk', 'large', 'diameter: 27.5 in cm', 'height: 19.3 cm', 'rim: 24 cm', 'if filled: 18.0 kg')
Warning: Could not place bucket ID 7 on the pallet.
Warning: Could not place bucket ID 6 on the pallet.
Do you approve the current stacking layout? (yes/no):
```

Figure 23: Details about each bucket included in an order is printed. A warning is also printed for the buckets not placed

If it is an instance of several orders being entered at once, this approval must be done before any following order plots are displayed. This is done by calling the get_confirmation_user function, which upon receiving an approval will return true. When returning true, the simulate_robot_actions is triggered to simulate the placements approved. A completion message will then be printed, and the break statement will be executed. This results in exiting the while true loop in the process_orders, enabling the process to move on to any following orders. Once all orders are processed the script will return to the main menu.

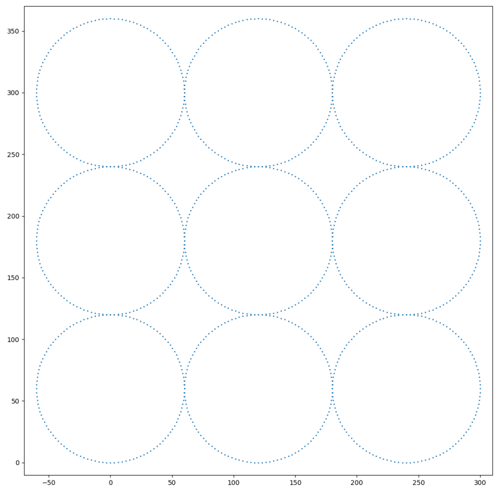
main_menu

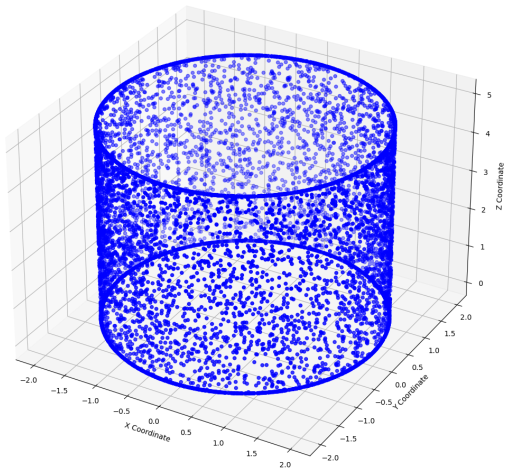
Upon running the script, the contents in this function is presented to the user. This contains the option of the main menu, and depending on the choices made will proceed to initialize different parts of the script. it consists of 3 options, input order, query about buckets in the database or to terminate the process. the menu is structured so that users can enter the numbers 1, 2 or 3 to access each part of the menu. Upon choosing a invalid integer higher than 3, the script will alert the user, and ask the user to input either option 1,2 or 3. Choosing option 1 and completing an order entry will print out the content in the order and the number of paint buckets it contains. Choosing 3 will immediately exit the script by breaking the loop.

Overall, the script takes care of connecting to an already established database containing logged information about paint buckets kept in stock. The script is based around a off-line and best fit approach as this was deemed most suitable for this objective. It is assumed that a proper log of inventory stock is kept in a warehouse, and that every item in an order is presumed to be known beforehand. This way the best possible layout of every order can be achieved. It was chosen to pick the heaviest buckets first for every order to establish a solid foundation for further stacking, so the best fit approach works by finding the best fit for each bucket in a decreasing size.

4.3.3 RANSAC Implementation

Acquiring test data: At the start of the year the group did not have access to a 3d camera that could operate with such large distances required by the project. therefore it became necessary to test and train using simulated point clouds. Initial trials was done on simple 2d plots in python matplot-lib library as seen in figure 24a below, to see if there was a simple way to approximate the bucket given to us by Westing, though after several tries only rudimentary cylinder shapes were produced and the data did not represent the subtle details like handles or ridges along the exterior walls as seen in figure 24b below. Since manually programming did not yield satisfactory results, the method shifted over to simulate a lidar camera using some form of modeling software.

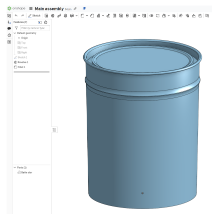

(a) 3x3 square of circles on 2d space

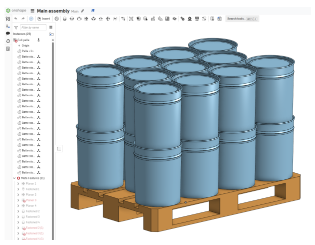


(b) Random generated cylinder in 3d space

Figure 24: Python based generation of pointcloud data

Before any data could be generated 3d models had to be made. The majority of all work was done in Onshape[27], where a detailed replica of the metal bucket supplied by Westing was placed, duplicated, stacked on a euro-pallet, and placed in warehouse shelving as seen in figure 25 a,b,c below.


(a) Recreation of bucket


(b) populated europallet

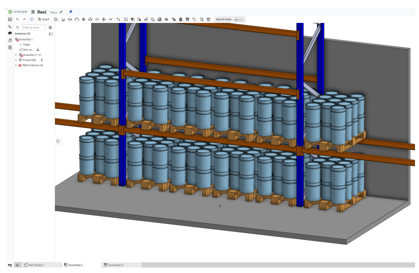
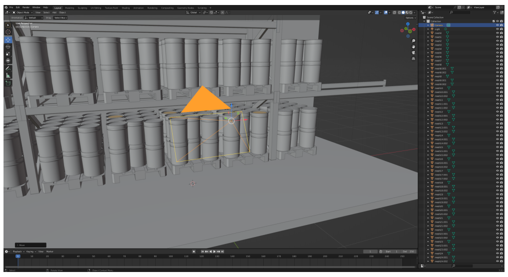
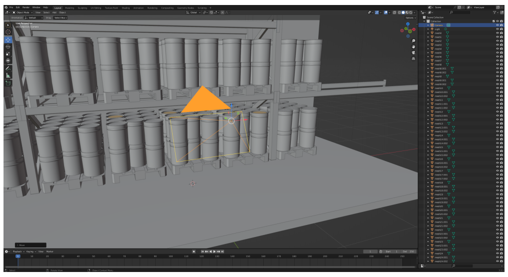

(c) Fully stacked shelves

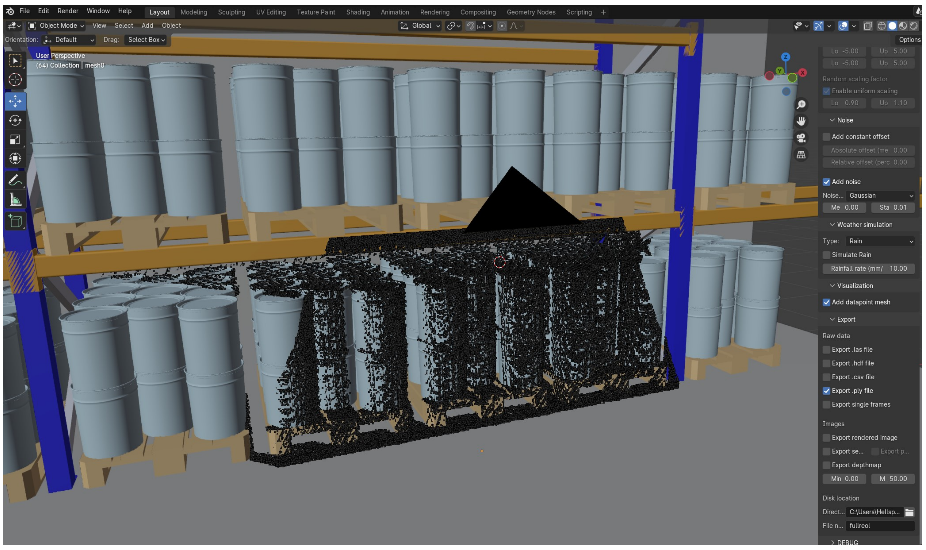
Figure 25: Generation of 3d models

After generating a fully stacked shelf with floor and wall it was loaded into Blender[28].

by installing a addon calles "blainder-range-scanner"[29], it is possible to use blenders camera object as a viritual lidar 3d camera and scan whatever is in front of it see figure 26 below. The generated point cloud data was then saved as a .ply file and loaded into python for further processing.


(a) The model is loaded into blender


(b) Generated pointcloud from camera perspective



Lo -5.00 Up 5.00
Lo -5.00 Up 5.00
Random scaling factor
[✓] Enable uniform scaling
Lo 0.90 Up 1.10
Noise
[ ] Add constant offset
Absolute offset (me 0.00
Relative offset (perc 0.00
[✓] Add noise
Noise... Gaussian
Me 0.00 Sta 0.01
Weather simulation
Type: Rain
[ ] Simulate Rain
Rainfall rate (mm/ 10.00
Visualization
[✓] Add datapoint mesh
Export
Raw data
[ ] Export .las file
[ ] Export .hdf file
[ ] Export .csv file
[✓] Export .ply file
[ ] Export single frames
Images
[ ] Export rendered image
[ ] Export se... [ ] Export p...
[ ] Export depthmap
Min 0.00 M 50.00
Disk location
Direct... C:\Users\Hellsp...
File n... fullreol
DEBUG

(c) Simulated lidar cam in blender

Figure 26: Blender workflow

RANSAC: The search algorithm is quite rudimentary and is based on the RANSAC algorithm[21]. The algorithm is a iterative method and works by the following steps or by looking at figure 27 below:

1. A random set of 3 different points is selected from the data set.
2. Using the X,Y coordinates of the 3 points a circle radius and center is calculated.
3. If the circle has a radius equal to the expected radius within tolerance, the circle is considered a potential bucket.
4. All other points in the data set have their distance to the centre of the circle calculated and if the distance is within a tolerance of the radius, the point is logged.

5. If the amount of innliers is above a certain threshold, the circle is considered a bucket and all innliers are removed from the data set.
6. The algorithm continues to search until its reached its maximum attempts (which is reset after every find), or there are too few remaining points to even reach the threshold.

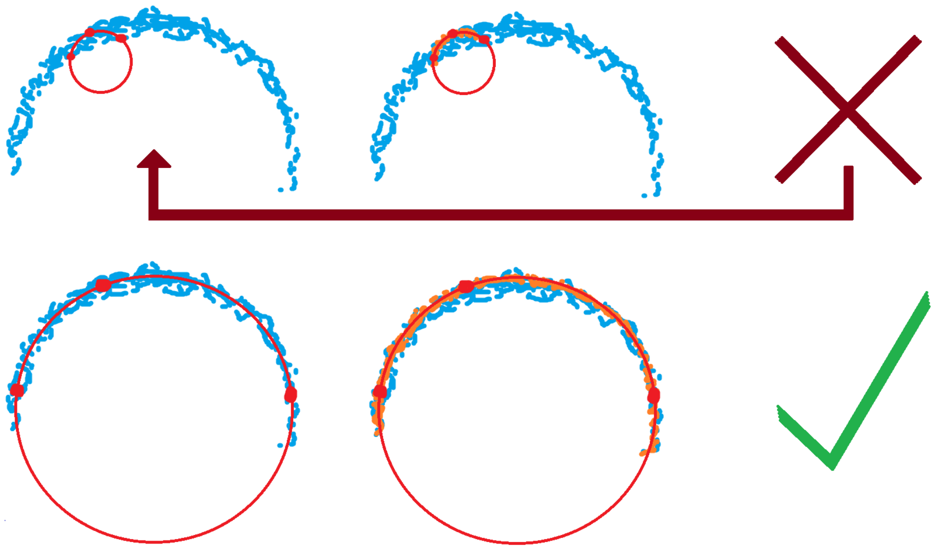
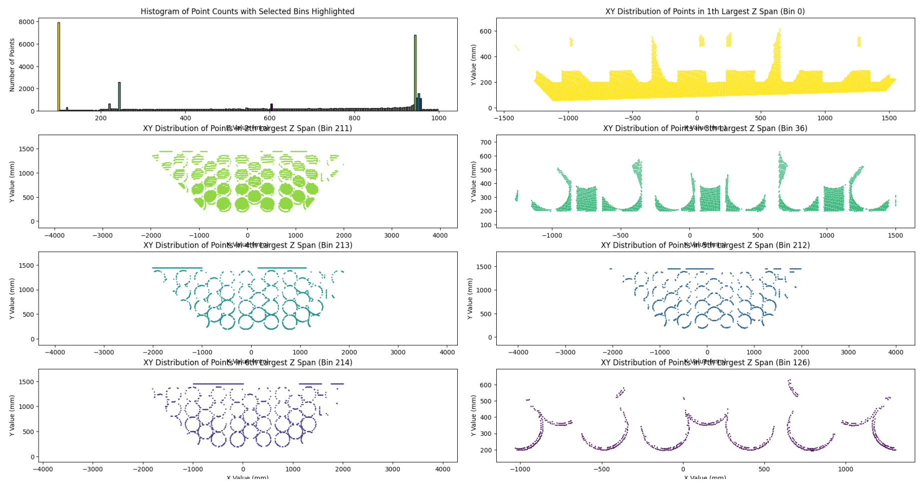
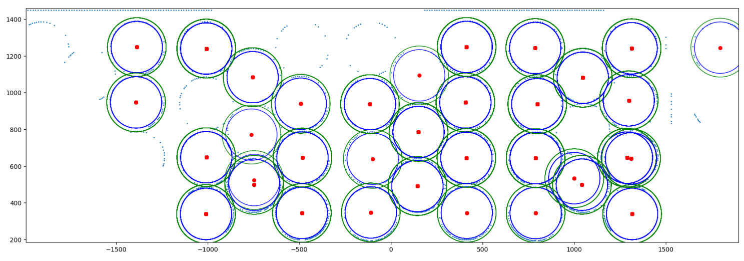


Figure 27: RANSAC graphically explained

Implementation: The point cloud data was then loaded into python and the RANSAC algorithm was integrated. At first a histogram was made to see the distribution of points vertically, and then the RANSAC algorithm was applied to the top n layers with the highest amount of points as seen in figure 28 below. The algorithm was able to detect the bucket and the floor with a high degree of but would not be a sustainable solution as depending on how many buckets are on a pallet and how they are stacked, it would become increasingly difficult to detect how many layers of buckets there are.



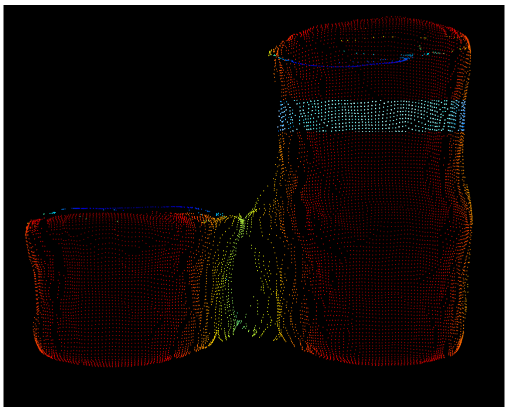
(a) Distribution of points and top 7 biggest layers

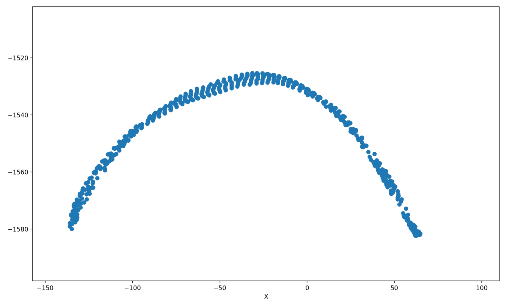


(b) RANSAC applied to Blender data

Figure 28: Early tests on data produced in blender

Thus a new approach was selected. As warehouses in which this system would be implemented keeps a single type of product on each pallet, the ransac algorithm can pull dimensional data about each product when searching. Allowing us to filter out a significant amount of data from each scan. The routine can then filter out all data other than a configurable height band at the centre of where the highest potential bucket should be. Additionally to simplify the remaining points, all Z coordinates are flattened to allow RANSAC to search along a 2d plane simplifying the process as seen in figure 29 below. Should no buckets be found at a expected height, it lowers the search by the products height and continues until the pallet is deemed empty. If multiple buckets are found in a single pass, it simply selects the one furthest towards the front and left for picking.


(a) Algorithm finds the highest bucket and selects points



(b) Curve from highlighted area

Figure 29: Resulting data that RANSAC searches through

4.3.4 Communication

For controlling both the linear axis as well as the Viper850 arm in manulab it was necessary to integrate our code on the existing architecture. Manulab runs its own network with central PLC's running Sysmac and has access to all equipment except the viper arm itself. The viper is controlled by a stripped down windows-machine running ACE, though still connected to the same network.

Python:
Both Sysmac and ACE are controlled by python scripts. The python script is run on the same machine as the camera and communicated over wifi using the python "opcua" library. This allows us to read and write data straight into variables stored on the PLC's. The script is run in a loop where it takes X,Y,Z picking coordinates as well as X,Y,Z placement coordinates from the packing algorithm. It then follows a sequence of movements alternating between ACE and Sysmac in order to ensure there is nothing in the path of the robot. Both ACE and Sysmac continuously reports the state of their given task to avoid any collisions.

Viper and ACE:
As ACE itself is an older software platform it only contained the older implementation of OPC called "OPC Data Access", this version works on establishing links across devices over COM port communication, making over-network connections extremely tedious and vulnerable security wise as to enable it, several changes to the windows firewall needs to be made. Though as the machine is still on the network a workaround was found using a software called "OPC Expert". This software was just lightweight and small enough to be installed side by side with ACE and acts as a data broker. OPC Expert sets up a virtual OPC server and passes through any data from the locally run OPC DA traffic coming from ACE.

Sysmac:
Sysmac uses the newer OPC UA standard, which is more secure and easier to work with. As Sysmac is only needed to move the linear axis that Viper rests upon, not

much data is needed to be passed between the two. The only data that is passed is the current "Y" position of the linear axis, and the current state of the robot. This is done by sending a positional coordinate along with velocity, and a final "Execute" boolean variable after a small delay to ensure data has been received properly. When any linear movement is done, the robot sends a "Done" signal back to the python script to indicate that the robot is ready for the next task.

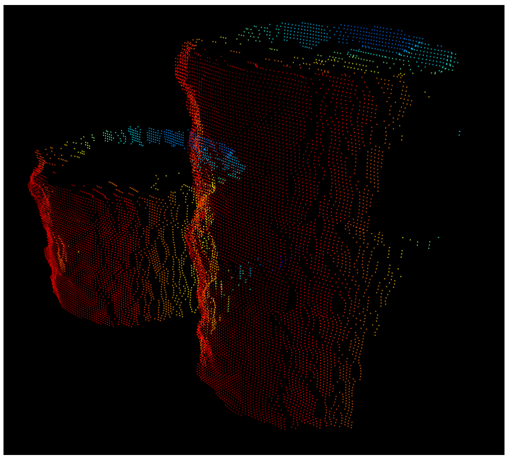
4.3.5 Filter

For filtering two methods became relevant. the first was to minimise the initial amount of data given by the camera. As most of the points are irrelevant or noise a filter was constructed to remove all points outside a given boundary The area is roughly defined from the top-left-front of a pallet to the top-right-rear of the storage rack. This was executed with a rudimentary python list comprehension function.

```python
def filter_point_cloud(points, min_coords, max_coords):
    # Create a mask for points within the specified range
    mask = (
        (points[:, 0] >= min_coords[0]) & (points[:, 0] <=
        ↪  max_coords[0]) &
        (points[:, 1] >= min_coords[1]) & (points[:, 1] <=
        ↪  max_coords[1]) &
        (points[:, 2] >= min_coords[2]) & (points[:, 2] <=
        ↪  max_coords[2])
    )
    # Apply the mask to the points
    filtered_points = points[mask]
    return filtered_points
```

The second filter was a bilateral filter to smooth out the stepping effect given by the intel realsense camera to allow for tighter tolerance in the RANSAC algorithm. The filter was implemented using open3d and numpy.

```python
def spatial_smoothing(points, radius=0.05, max_neighbors=30):
    pcd = o3d.geometry.PointCloud()
    pcd.points = o3d.utility.Vector3dVector(points)
    pcd_tree = o3d.geometry.KDTreeFlann(pcd)
    filtered_points = np.zeros_like(points)
    for i in range(len(points)):
        [k, idx, _] = pcd_tree.search_radius_vector_3d(pcd.points[i],
        ↪  radius)
        if k > 0:
            # Limit the number of neighbors to max_neighbors if
            ↪  there are more found
            effective_neighbors = idx[:min(k, max_neighbors)]
            filtered_points[i] =
            ↪   np.mean(np.asarray(pcd.points)[effective_neighbors],
            ↪   axis=0)
        else:
            filtered_points[i] = points[i]
    return filtered_points
```


(a) Unfiltered pallet space

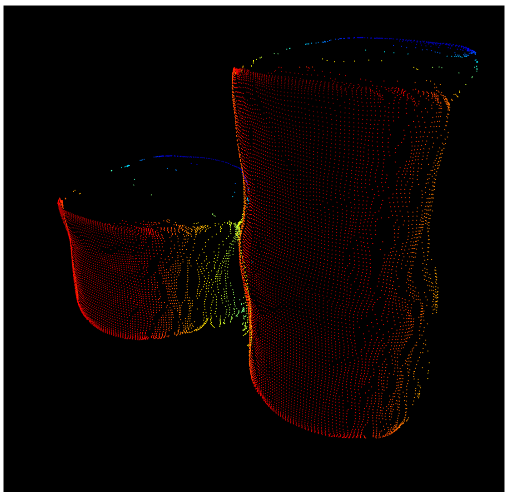

(b) Filtered pallet space

Figure 30: Filtering of pointcloud data

4.3.6 Main

To tie all the sub components together, all previous systems were imported into one main program so that any changes to parameters could easily be altered. These variables ranged from RANSAC search values to filtering coordinates and OPC node id's. Further the program also initialises the realsence camera for continuous monitoring and pulling the latest image before a scan. This initialization also runs in a parallel thread to avoid having the camera and ransac running at the same time causing delays.

The complete script works as follows:

1. On startup intel realsence camera gets initialized
2. The packing algorithm UI gets called and awaits input.
3. When a list of buckets have been assigned a packing plan it feeds off the bucket measurements into the main loop
4. The latest point cloud captured by the camera gets filtered out.
5. The program searches for any points along the heights the sought after bucket could inhabit.

6. RANSAC algorithm gets applied to a 2d plane from the search band.
7. When a bucket gets found, the coordinates gets sent along with the placement destination to the communications script.
8. Python handles the step by step process for movements between ACE and Sysmac, figure: 31.
9. After a completed pick and place routine the process gets repeated until the order is complete.
10. The initial UI gets called again for new orders.

4.4 Integrating ACE, Sysmac and Python via OPC

The integration of OPC communication was essential for controlling both the linear axis and the robotic arm. This was accomplished by integrating our code with the existing architecture at Manulab, which operates on a network with central PLCs running Sysmac and has access to all equipment except the Viper arm. The Viper is managed by a dedicated Windows machine running ACE, which is still connected to the same network.

Both Sysmac and ACE are controlled by Python scripts. The Python script, running on the same machine as the camera, uses the opcua library for communication over Wi-Fi. This setup allows direct reading and writing of data into variables stored on the PLCs. The script operates in a loop, processing X, Y, Z coordinates for picking and placing tasks. It ensures the robot's path is clear by alternating control between ACE and Sysmac, which continuously report the status of their tasks to avoid collisions.

ACE, an older software platform, uses the OPC Data Access (OPC DA) standard, which relies on COM port communication. This method of communication can be complex and insecure due to necessary firewall modifications. To facilitate over-network communication, OPC Expert was employed as a lightweight intermediary software. OPC Expert establishes a virtual OPC server that bridges local OPC DA traffic from ACE to the network.

However, due to the free version's limitations, OPC Expert required restarting every four hours. Despite this, the software proved effective in maintaining reliable communication between ACE and the network.

Sysmac uses the newer and more secure OPC UA standard. This standard simplifies communication and enhances security. For controlling the linear axis of the Viper, minimal data exchange is required. The necessary data includes the current ”Y” position and the robot's status. Positional coordinates and velocity are sent to Sysmac, followed by an ”Execute” boolean variable to confirm data reception. Upon completing any linear movement, Sysmac sends a ”Done” signal back to the Python script, indicating readiness for the next task.

Figure 31 represents a swim lane diagram that illustrates the integration workflow among Sysmac, ACE, and Python via OPC UA communication. Python sends coordinates for pick and place operations to ACE and Sysmac. ACE and Sysmac set specific variables to signal when Python should proceed with sending these coordinates. This process operates in a continuous loop, restarting once the cycle is completed to ensure ongoing and synchronized operations across the different systems.

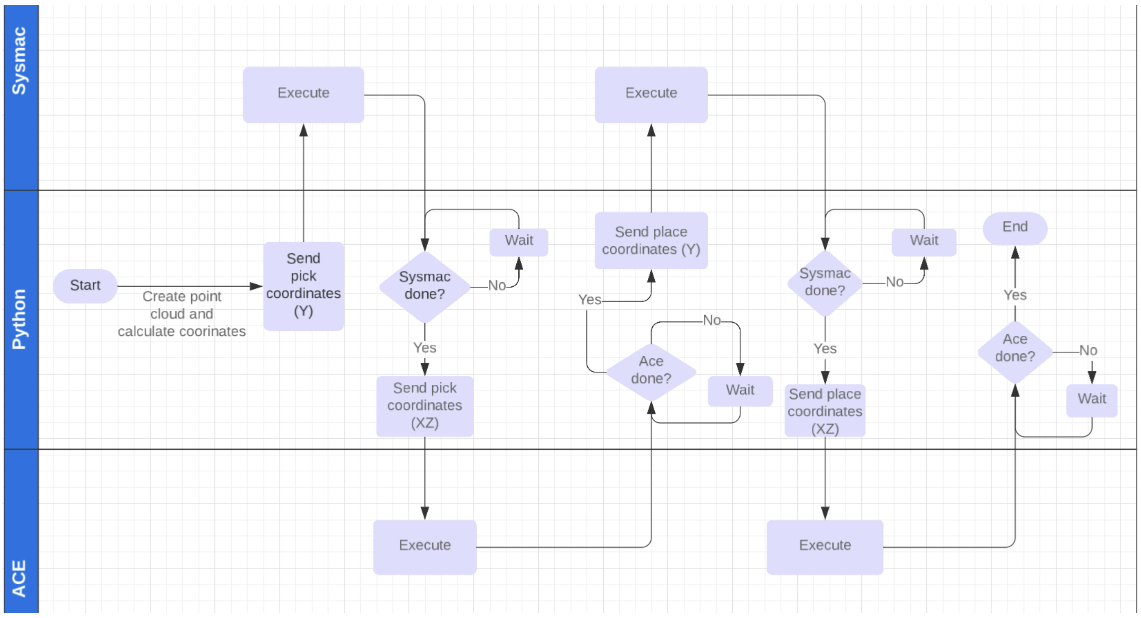

Sysmac
Python
ACE
Execute
Execute
Start
Create point cloud and calculate coorinates
Send pick coordinates (Y)
Sysmac done?
No
Wait
Yes
Send pick coordinates (XZ)
Execute
Send place coordinates (Y)
Yes
Ace done?
No
Wait
Sysmac done?
No
Wait
Yes
Send place coordinates (XZ)
Execute
End
Yes
Ace done?
No
Wait

Figure 31: Swim lane diagram: ACE, Sysmac and Python

4.5 Sysmac Program

As illustrated in figure 32, the Sysmac program dictates the linear motion paths in the y direction for the robot. This was implemented to allow the robot to mimic how Grab drives to the pallet of buckets before performing the pick action.

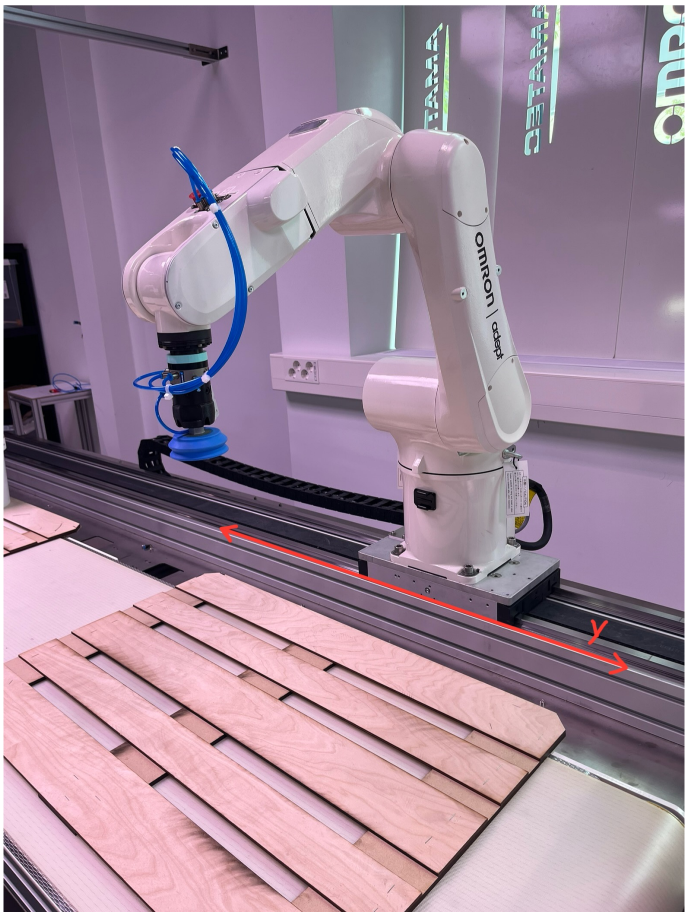


Figure 32: Sysmac - robot direction

The robot is configured to navigate between coordinates -1050 and 1600 safely, eliminating the risk of colliding with the conveyor belt's boundaries. This configuration gives it a movement range of approximately 3 meters.

The program combines Ladder logic and Function blocks, receiving both the linear position and velocity from the Python program. It waits for the "ExecutePick" command before driving the robot to the specified position. Once the robot reaches its destination, it sends a "Done" signal to the Python script, allowing the script to proceed with the next steps.

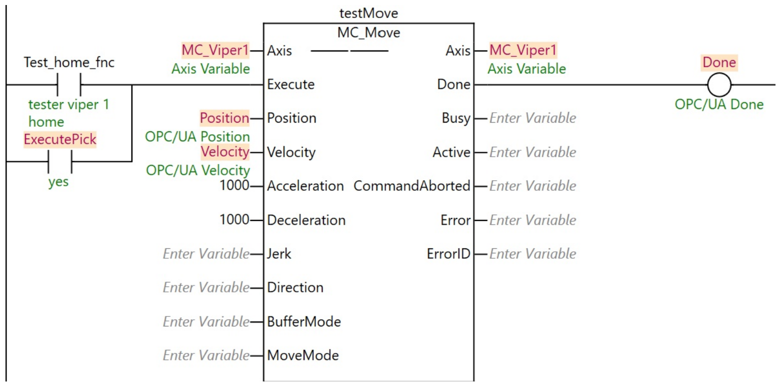


Figure 33: Sysmac program

4.6 ACE Program

After Sysmac is done driving the robot to the specified position, the ACE program is excecuted. This program directs the robotic arm's x and z movements, as illustrated in figure 34.

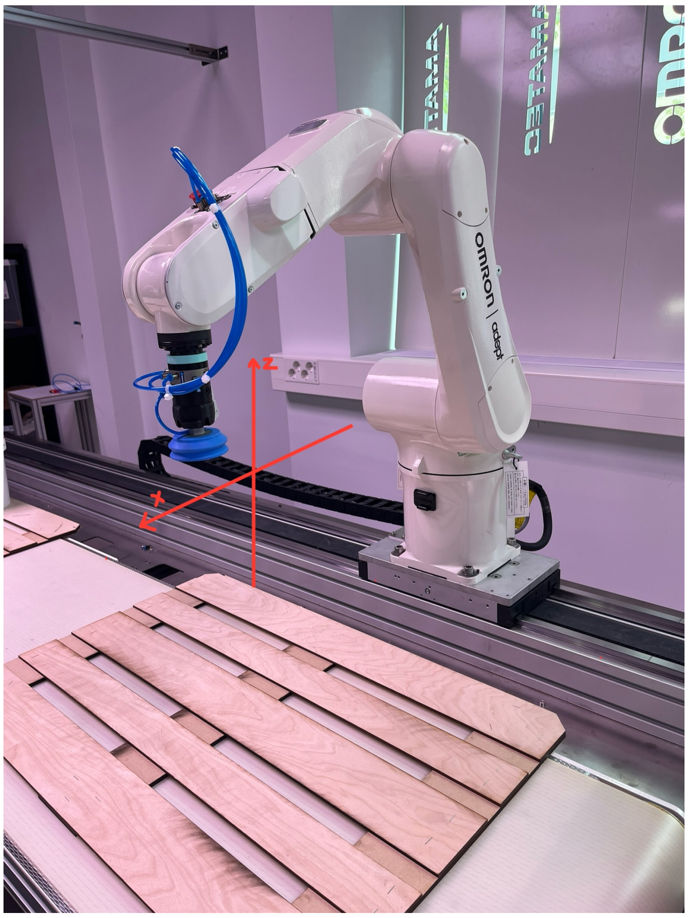


Figure 34: ACE - robot direction

It operates with a simple loop containing two if statements that run indefinitely. The first if statement manages the pick task, while the second handles the place task. The loop begins when the "pickready" variable is set to 1, indicating a bucket is ready for pickup. The robot aligns with the bucket's coordinates provided by the Python program, activates its suction mechanism, and moves the bucket to new coordinates, placing it on a pallet. Once completed, the robot returns to its home position, ready for the next set of coordinates.

The "neutral" position ensures the robot avoids collisions while picking and stacking the buckets.

```
.PROGRAM program0()
    SET home = TRANS(125, 0.1, 744, 0, 180, 0)
    MOVE home
    BREAK
```

```
pickready = 0
pickdone = 0
placeready = 0
placedone = 0

DO
    IF pickready == 1 THEN
        pickready = 0

        SET neutral = TRANS(xpos, 0, 600, 0, 180, 0)
        MOVES neutral
        BREAK

        SET pick = TRANS(xpos, 0, zpos, 0, 180, 0)
        MOVES pick
        BREAK
        SIGNAL 5
        WAIT.EVENT, 0.5
        BREAK

        SET neutral = TRANS(xpos, 0, 600, 0, 180, 0)
        MOVES neutral
        BREAK

        SET neutral = TRANS(destx, 0, 600, 0, 180, 0)
        MOVES neutral
        BREAK

        pickdone = 1
        WAIT.EVENT, 2
        pickdone = 0

    ELSE
        BREAK
    END

    IF placeready == 1 THEN
        placeready = 0

        SET place = TRANS(destx, 0, destz, 0, 180, 0)
        MOVES place
        BREAK
        SIGNAL -5
        SIGNAL 6
        WAIT.EVENT, 0.2
        SIGNAL -6
        WAIT.EVENT, 0.2
```

```
            BREAK

            SET neutral = TRANS(destx, 0, 600, 0, 180, 0)
            MOVES neutral
            BREAK

            placedone = 1
            MOVE home
            BREAK
            placedone = 0

        ELSE
            BREAK
        END

    UNTIL FALSE
.END
```

4.7 Suction cup

The chosen suction cup is made of rubber and is bellow, which means that it have an accordion-like structure that allows it to compress and extend, providing flexibility and adaptability in gripping objects of various shapes and sizes. The bellows design enables the suction cup to create a better seal on uneven or textured surfaces, improving the overall gripping performance of the robotic gripper [30].

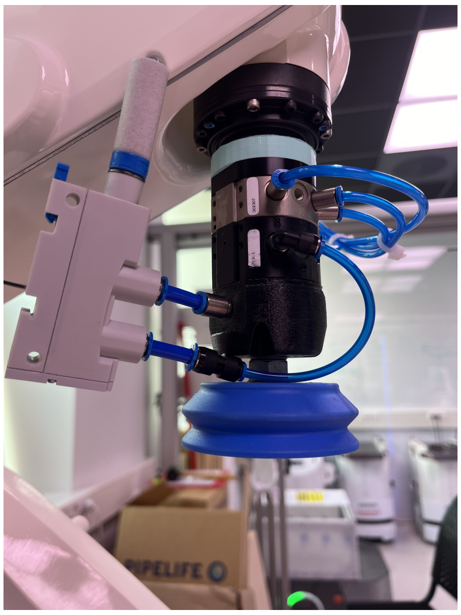


Figure 35: The chosen suction cup

5 Result

This section will present the obtained results during the project. This includes controlling the robotic arm, and the programs written to do so. Communication is also described in the OPC subsection. Following this an explanation of the paint bucket and packing algorithm in Python is detailed. The result section is concluded with an overview of how the Ransac algorithm performed.

5.1 Work Process

Initially, we planned to have meetings with our supervisor once every week or two. However, this schedule had to be adjusted as some weeks required no supervision, while others required more frequent guidance. During these meetings, we presented our progress and sought assistance as necessary.

During the early stages, progress was slow. Initially, we tried constructing our own TCP packages, but this was quickly established to be a dead end as we could not send values higher than 128. Finding a communication method that could satisfy all three softwares was difficult, as ACE lacked effective interface methods. This led to many days spent researching alternate methods before ultimately settling on OPC.

5.2 System Overview

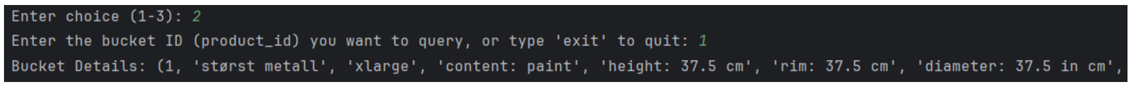
When it comes to our system as a whole, it has been working as expected the whole time. While there was a couple of times we needed to restart a computer or PLC due to uncontrollable external factors, everything worked as intended. One important note is that once each day, OPC Expert needs to be restarted as it is a trial version that can only run continuously for 4 hours before shutting down.

5.3 Python

While the python scripts usually worked as intended, there would be the occasional error and crash. Some times empty camera frames would lead to exceptions and crashes as not all functions had error handling. These fault usually showed up when the camera started acting explainable like shifting coordinates or just outputting empty data.

5.3.1 Paint Bucket Database

The database stored all the information needed about the buckets, connecting to it lets the user retrieve anything of interest. The database works with the packing algorithm to produce plots of the orders, and printing detailed information about the buckets. Figure 36 illustrates information retrieved about a bucket, printed for the user.

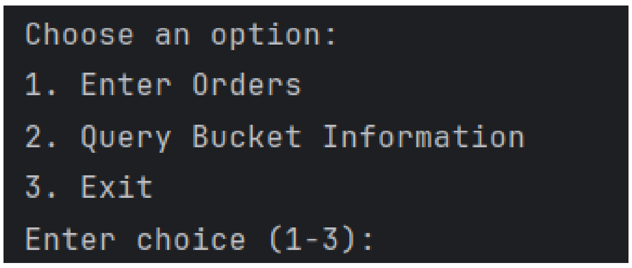


```
Enter choice (1-3): 2
Enter the bucket ID (product_id) you want to query, or type 'exit' to quit: 1
Bucket Details: (1, 'størst metall', 'xlarge', 'content: paint', 'height: 37.5 cm', 'rim: 37.5 cm', 'diameter: 37.5 in cm',
```

Figure 36: Extracting information about bucket 1

5.3.2 Packing Algorithm Script

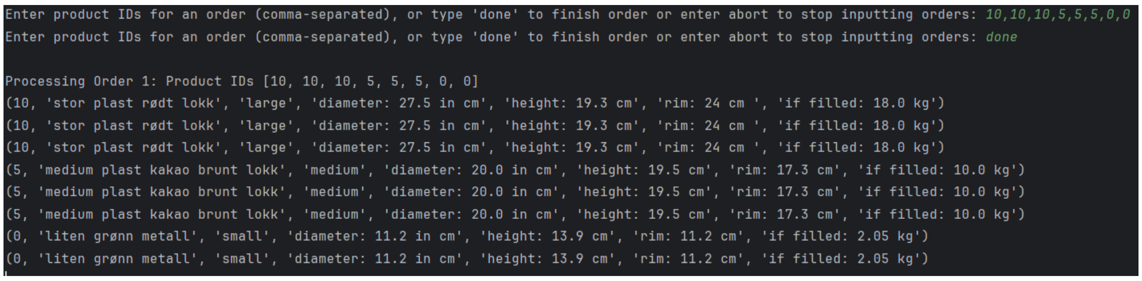
The packing algorithm handles communication with the database, and order generated from user input. It handles creating plots of orders and querying information about buckets from the user. The menu is designed to contain concrete choices, including information about how to navigate the menu. Type 1 for entering orders or 2 for extracting information about specific IDs. A third option was later added to exit the code by pressing the 3 key. The menu is shown in figure 37.



```
Choose an option:
1. Enter Orders
2. Query Bucket Information
3. Exit
Enter choice (1-3):
```

Figure 37: A screenshot showcasing the menu choices

Illustrating the usage and results of the algorithm could be performed via an example. In this example, personnel has ordered 3 large-, 3 medium- and 2 small buckets. The user first has to type 1 to enter orders, then type the bucket IDs of the respective sizes as shown in figure 38. The ID 10 is assosiated with a large bucket, ID 5 is assosiated with a medium and ID 0 is a small bucket.



```
Enter product IDs for an order (comma-separated), or type 'done' to finish order or enter abort to stop inputting orders: 10,10,10,5,5,5,0,0
Enter product IDs for an order (comma-separated), or type 'done' to finish order or enter abort to stop inputting orders: done

Processing Order 1: Product IDs [10, 10, 10, 5, 5, 5, 0, 0]
(10, 'stor plast rødt lokk', 'large', 'diameter: 27.5 in cm', 'height: 19.3 cm', 'rim: 24 cm ', 'if filled: 18.0 kg')
(10, 'stor plast rødt lokk', 'large', 'diameter: 27.5 in cm', 'height: 19.3 cm', 'rim: 24 cm ', 'if filled: 18.0 kg')
(10, 'stor plast rødt lokk', 'large', 'diameter: 27.5 in cm', 'height: 19.3 cm', 'rim: 24 cm ', 'if filled: 18.0 kg')
(5, 'medium plast kakao brunt lokk', 'medium', 'diameter: 20.0 in cm', 'height: 19.5 cm', 'rim: 17.3 cm', 'if filled: 10.0 kg')
(5, 'medium plast kakao brunt lokk', 'medium', 'diameter: 20.0 in cm', 'height: 19.5 cm', 'rim: 17.3 cm', 'if filled: 10.0 kg')
(5, 'medium plast kakao brunt lokk', 'medium', 'diameter: 20.0 in cm', 'height: 19.5 cm', 'rim: 17.3 cm', 'if filled: 10.0 kg')
(0, 'liten grønn metall', 'small', 'diameter: 11.2 in cm', 'height: 13.9 cm', 'rim: 11.2 cm', 'if filled: 2.05 kg')
(0, 'liten grønn metall', 'small', 'diameter: 11.2 in cm', 'height: 13.9 cm', 'rim: 11.2 cm', 'if filled: 2.05 kg')
```

Figure 38: Example of order from warehouse personell with 3 large-, 3 medium- and 2 small buckets

The algorithm then attempts to optimally place the ordered buckets and shows a plot of their location. In this example, the plot in figure 39 is presented.

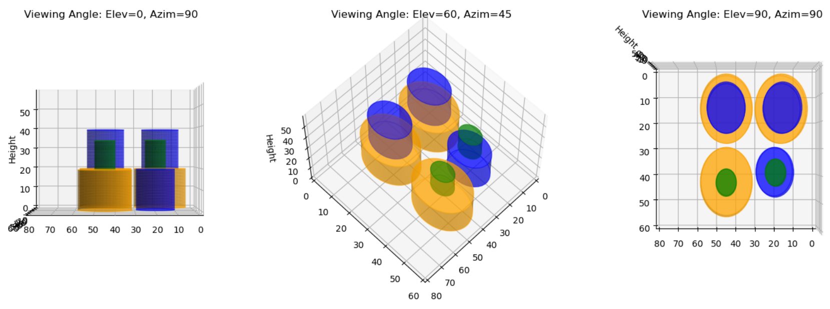

Viewing Angle: Elev=0, Azim=90
Viewing Angle: Elev=60, Azim=45
Viewing Angle: Elev=90, Azim=90
Height

Figure 39: Illustrated bucket placements from example order

If the placements are acceptable, the user can type "done" to begin stacking and the camera will attempt to locate the entered buckets.

One thing to note is that more buckets in the order lead to longer processing time. This delay is insignificant, as it is at most some seconds added.

5.3.3 RANSAC Implementation

Detection:
When it came to detection of paint buckets there was never any real issues related to our implementation of RANSAC. Even before the raw point clouds were filtered, RANSAC proved able to detect and properly detect the right sized buckets. There was however a large degree of inaccuracy when finding the centre of the bucket. This was caused by the stepping effect from the camera, leading to a higher needed tolerance in order to get enough points to count as a find. The dilemma can be summarised as such:

1. Without filtering the overall inner and outer curve of points could produce a 10mm thick ring forcing a higher tolerance.
2. With higher tolerance on RANSAC, more plausible circle placements are introduced, increasing inaccuracies.
3. Lowering tolerance without filtering means the threshold limit would also need to be lowered, increasing chances of false positives.

Early testing on clean generated data proved not an issue however. While the first real world tests needed a tolerance of 10-20mm, using simulated data buckets could be detected as low as sub 0.5mm but at the cost of increased search time, see figure 40 below.

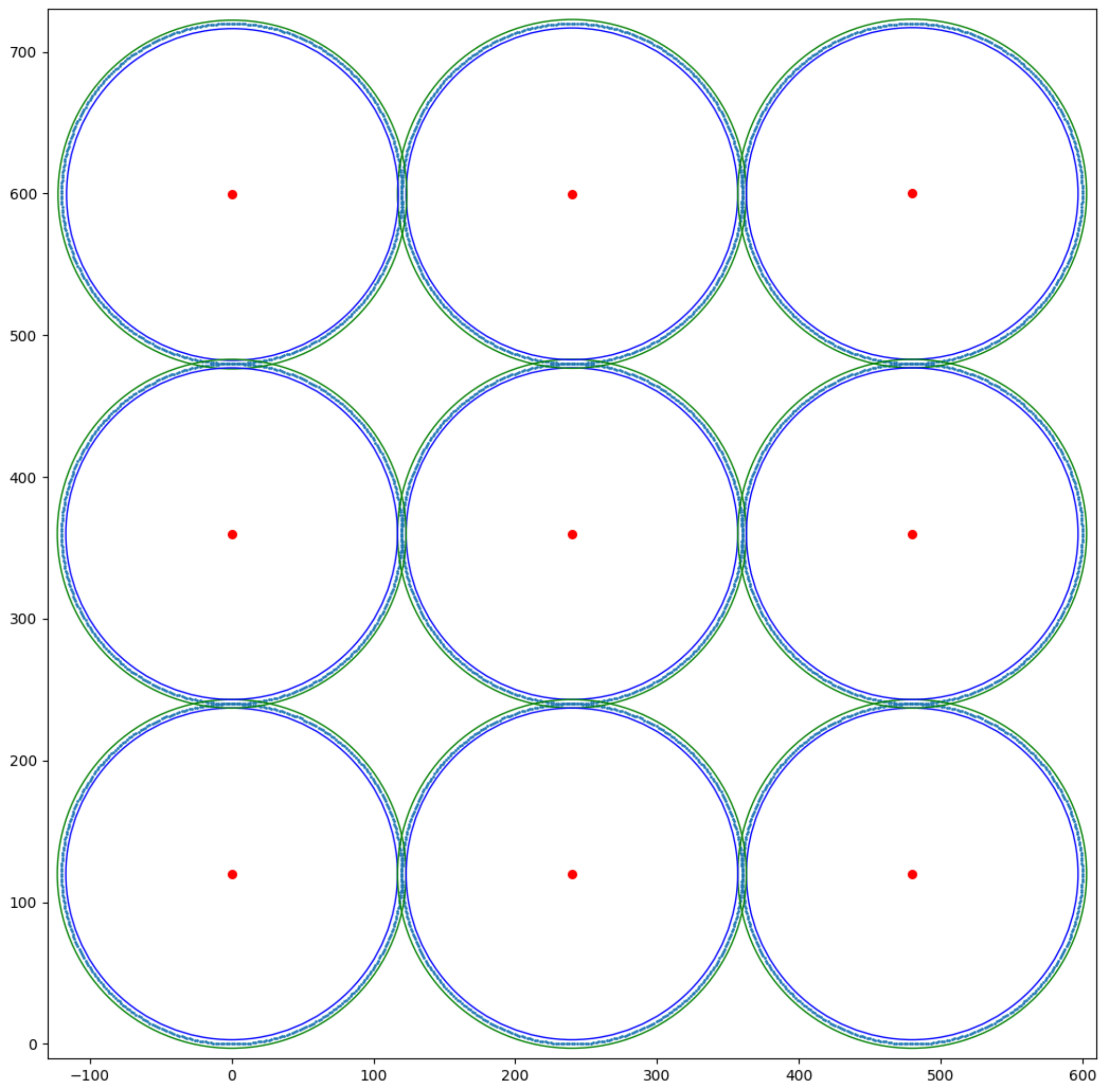


Figure 40: First test using RANSAC on simulated data

A potential scenario to account for is when two objects have overlapping data points inside overlapping tolerance areas see figure 41 below. As the algorithm removes points on a successfully find, you also loose points belonging to other objects. This can quickly be mitigated by having a lower tolerance or lowering the point threshold.

Accuracy:
While the algorithm would almost always find desired buckets, the ability to find the exact centre could vary a lot depending on circumstances.

Throughout the project several attempts were made to test accuracy, In order to minimise disturbances each test was taken by placing a single bucket within detection distance while RANSAC was applied several times in a row. These results are the various co-ordinates along the x axis the buckets centre was detected. Therefore the size of the coordinates are not important, rather the spread of each measurement. Coordinates will also shift as these tests were not taken on the same day but as the project progressed throughout the year.

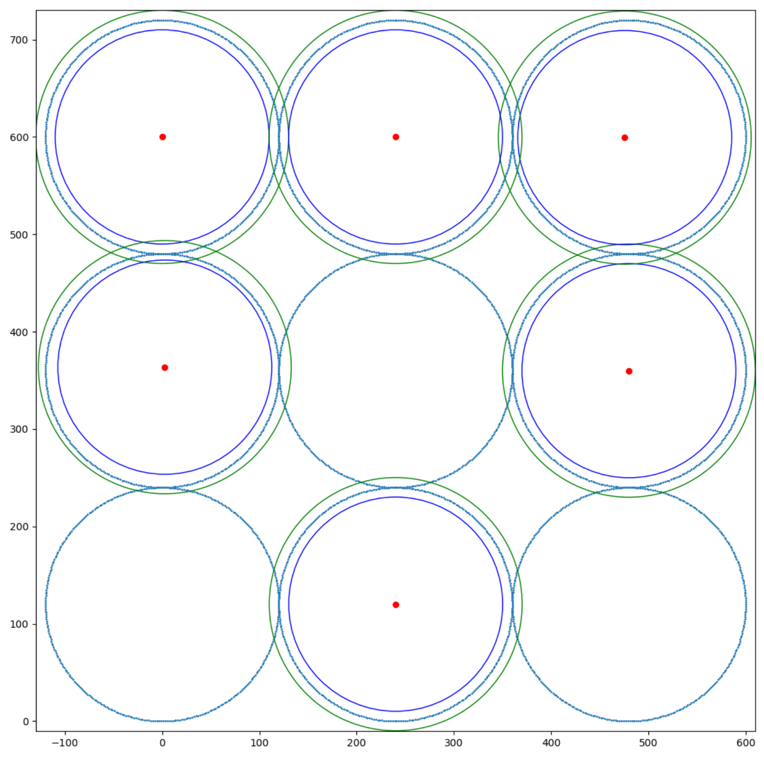


Figure 41: Corner and middle circles not found due to not enough points

Another important distinction is there had been lapses of problems using the L515 camera. At seemingly random intervals coordinates in the incoming point clouds could shift with anything from +-20cm on X axis to 1m straight up on Z axis, the cause was never found and usually resolved itself between restarts or being left unplugged to cool down. The group was able to acquire another camera towards the end of the project so its important to note that the first two columns of tests were done with the problematic camera and the third one with the new one (table: 7). Both cameras also produced a weird behavior that when initialising each time, coordinates would shift even when nothing had been physically moved. The assumed cause was improper laser alignment between power cycles leading to the cloud being rotated side to side. This would also support the fact that points rarely ever deviated in depth from the camera perspective, just shifted side to side or up and down. This got fixed before the third test by letting the camera stay initialised throughout the picking process and grabbing the latest frame when detecting, and produced very satisfactory results. It did still however cause buckets to be stacked slightly off the destination each run, however all buckets had an almost identical offset so for a per order basis this worked out fine.

No filter and camera reboot Tolerance 17mm Old camera	With filter and reboot Tolerance 10mm Old camera	Instanced camera and filter Tolerance 2mm New camera
35	18	9
22	15	9
28	12	7
27	13	6
21	10	8
14	10	5
32	11	7
1	11	6
-12	9	3
-24	6	9
26	5	8
5	5	8
30	-6	4
-28	13	5
Max variance 63mm	Max variance 24mm	Max variance 6mm

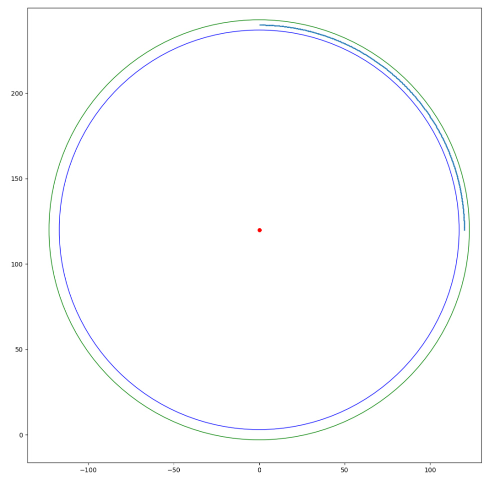
Table 7: Repeated static RANSAC testing with result in ACE X coordinates.

5.3.4 Communication

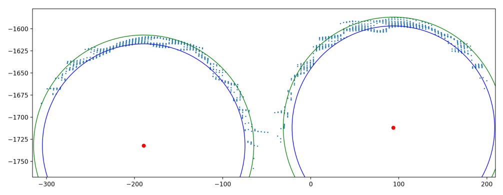
Upon discovering the OPC library for Python, the communication between the PLC and Python proceeded smoothly for the most part. The primary challenge involved learning the specific server URL format and identifying the node IDs for all shared variables. However, once these were correctly configured, there were no further issues with communication from Python.

5.3.5 Filter

The filtering process was a bit of a mixed bag. While it did remove a lot of noise from the point cloud, it also ended up smoothing away a lot of edges and details. Our filter implemented a bilateral filter, which is meant to preserve boundaries while smoothing, but due to the camera stepping effect, all smoothing parameters had to be increased to a point where it resembled a more traditional Gaussian filter. This was a necessary evil as the stepping effect would otherwise so dispersed points that tolerance would have to be unreasonable high see figure 42b below
Although the bilateral filter smoothed out the point cloud data to the point where the general curve was flattened on the transition between stacked buckets and buckets placed next to each other, it did not have any detrimental effects on the algorithm's detection ability. Since each bucket gets scanned in the middle, any smoothing vertically does not tamper with overall radius enough to cause issue. Also by adjusting the height of the search band the number of sample points can be increased to offset any loss of curve as ransac only needs a small portion of a circle and enough points to make a good estimate, see figure 42a below.



(a) Only a smaller curve is needed to achieve a find



(b) Rougher point clouds only needs larger tolerance

5.3.6 Main

Our main program performed its tasks efficiently. By separating the threads for the main process and the camera object, we achieved both fast code execution and precise operations. This approach ensured that the camera remained continuously active, enhancing the accuracy of our pickings.

5.4 Integrating ACE, Sysmac and Python via OPC

The integration of Sysmac, ACE and Python via OPC UA communication was successfully implemented, demonstrating efficient and synchronized operations for pick and place tasks in a robotic setup.

5.5 Sysmac Program

The performance of the Sysmac program was validated through consistent and accurate navigation of the robot. The implemented logic ensured reliable execution of the pick tasks, contributing to the overall efficiency of the workflow.

5.6 ACE Program

Following the completion of the Sysmac-driven tasks, the ACE program is executed. It effectively coordinated the pick and place process. The robot accurately aligned with the bucket's coordinates provided by Python, used its suction mechanism to pick up the bucket, and placed it at the new coordinates on the pallet. The continuous loop structure allowed for uninterrupted operation. However, due to there not being a vacuum monitor installed, the program will still continue with executing the pick and place task, even if a bucket is missed. This is a flaw that could easily be fixed by installing a vacuum monitor.

5.7 Suction cup

As our project is a scale model working on empty buckets there was never any problems lifting targets.

5.8 Accidents

As this project was an iterative task, we did encounter a few accidents while troubleshooting and integrating as seen below in figure 43 and 44. There were a couple of incidents were the robot would ram the bucket due to wrong coordinates. This was caused by wrong offsets being applied from pointcloud coordinates to Sysmac/ACE coordinates.

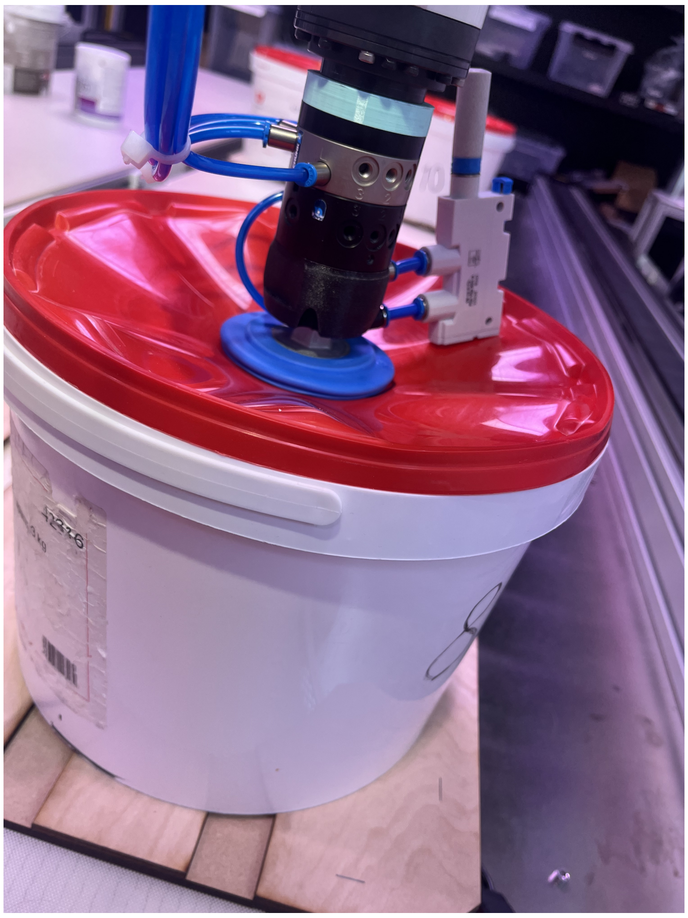


Figure 43: Misaligned coordinates can cause accident

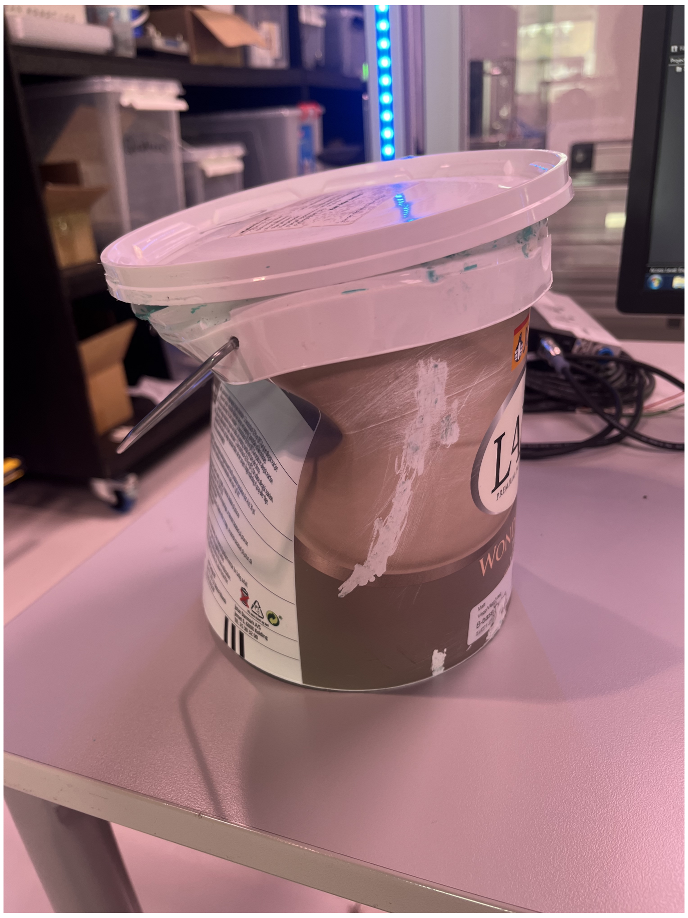


Figure 44: Deformed bucket due to wrongfully picking

6 Discussion

In this section some of the parts of this bachelor's thesis is further discussed. This includes what could have been done differently and the experiences gathered throughout this project. This includes discussing some of the hardware used, such as the suction cup and the paint buckets. It also discusses some of the software used and developed, and the issues had with the developed code.

6.1 Work Process

In the end, the group's work progressed as planned though at a slightly longer timescale. Although delays occurred due to OPC communication troubleshooting progress was steady. While we were not entirely able to stick to our schedule, we were able to complete all major deliverables. The one thing planned that was dropped was the QR code scanner as it was planned as an quality control feature if sufficient time was to spare.

6.2 Python

From the outset, the group agreed that Python would be the most optimal language for this project. Although Python may not be the most efficient language, it was the one we were most familiar with, giving us the best chance of producing a working prototype. However, it is important to note that algorithmic code, such as RANSAC detection, can greatly benefit from being executed in a more optimized language such as C++ or Rust.

6.2.1 Paint Bucket Database

SQLite was mainly chosen for the database because it is easy to understand and learn, and the module contained the functionality suitable for this objective. The SQLite3 module has a lot of documentation available, is reliable and changes can be done quickly and efficiently.

The database contains 19 slots containing information about the paint buckets. The barcode field is excessive, as the information stored in it was never used. The field max_stack_height creates a bit of uncertainty, as the limit is set to 2 for each paint bucket. This is only a generalization used in this project, and not a limitation.

6.2.2 Packing Algorithm Script

One thing to note is that currently paint buckets placed on top of others, are specifically placed at the centre. This was decided so that stable stacking was ensured. However, if the goal is to place as many buckets on a pallet as possible, this would not be optimal. If for instance a large bucket is placed on the bottom layer, and small buckets are to be placed on top, the large bucket could most likely fit multiple small. This is only the case for this specific scenario, but it is an improvement that could have been made.

The user interface could also have been improved. Currently when entering the bucket IDs the user has to manually enter all desired buckets. An improvement to this could have been to also present a way to enter the amount of buckets of the specified ID. This way the entering of the buckets would be less tedious and quicker.

6.2.3 RANSAC Implementation

The implementation of the RANSAC in our system yielded promising results, particularly in the detection of paint buckets. Despite initial challenges with noisy data, the algorithm demonstrated a robust ability to identify and differentiate between objects within the point cloud. However, several areas for improvement were identified throughout the testing phases.

While the RANSAC algorithm is able to provide results from noisy data, its current state still suffers from accuracy issues. A potential improvement could involve optimizing the inlier detection process. By increasing the tolerance initially, the algorithm could identify more consensus data to refine coordinates from. Once a circle is detected and verified, the center could be further processed by optimizing the distances of all inlier points to achieve an equal distance from the circle edge. This approach could enhance the precision of the detected circle and improve the overall accuracy of the system.

Another solution could be to keep our algorithm as is, while adding a normal camera looking up to grab. By adding a calibration between pick and place we can do edge detection looking at the underside of current picked bucket and determine a new centre from 2d images alone.

6.2.4 Communication

Initially, we used a TCP/IP server/client setup to send camera coordinates from Python to ACE. However, we later found that using OPC UA/DA was a simpler and more efficient solution for directly modifying variables, as it was also compatible with all programs used in this project.

6.2.5 Filter

It became apparent that using a bilateral filter was not the optimal one to use with our camera. The L515 produced so choppy depth measurements that any edge between stacked buckets would get caught in filtering as the stepping was so hard it couldn't be distinguished from transition between buckets as seen in figure 30b. To further improve filtering would either require more cameras so we can overlap and correlate data or do further research into other filtering methods.

6.2.6 Main

As far as our main program it all worked as intended, though two factors could be improved. Using a command line interface for requisitioning orders can be quite cumbersome and should have an accompanying graphical user interface. Another potential for improvement is further use of multi threading to divide the load during search, or adapt the code to run on the GPU rather than the CPU using a library like CuPy [31].

6.3 Integrating ACE, Sysmac and Python via OPC

The integrated workflow of Sysmac, ACE, and Python demonstrated a robust and synchronized performance, effectively handling the pick and place operations in a continuous loop. The results highlight the success of the integration and the reliability of the developed programs in achieving the desired results.

6.4 Sysmac and ACE Program

The performance of both programs was satisfactory. As for ACE, the robot consistently returned to its home position after each cycle, ready for subsequent tasks. For Sysmac, the robot consistently drove to the specified location. The programs maintained a high level of operational efficiency.

6.5 Suction cup

The chosen suction cup was effective for all bucket sizes except the smallest. It has a diameter of 7 cm, the same size as the pick area of the smallest bucket. The problem occurred because the robot could miss by a few millimeters, causing part of the suction cup to land in a small dent on the outer ring of the bucket, which resulted in a suction failure.

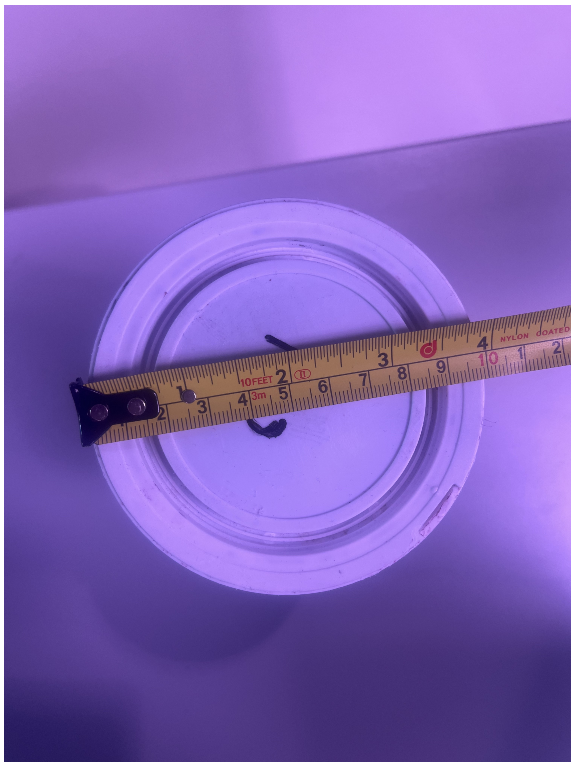


Figure 45: Smallest bucket

This could be solved with increased accuracy or investing in a slightly smaller suction cup.

Another challenge is that during stress-testing, it was found that the suction cup loses its grip with weights over 15 kg. This problem can be mitigated by using two or more suction cups or opting for a wider suction cup. However, this solution introduces new challenges, as picking up the smallest bucket requires a single suction cup with a maximum diameter of 7 cm or smaller. Therefore, if the goal is to pick up all bucket sizes using a single suction cup without switching between different sizes, it is advisable to select a different type of suction cup.

6.6 Vacuum

Currently, the system lacks a vacuum monitor, causing the robot to continue the picking and placing process even if it misses a bucket. To address this, a vacuum monitor should be installed to ensure the robot's program halts when it fails to pick up a bucket. With the monitor in place and properly integrated into the script, the robot can retry the action in the event of a vacuum failure during the picking process, preventing it from proceeding with stacking without a bucket.

This can easily be installed using the pins on top of the robot:

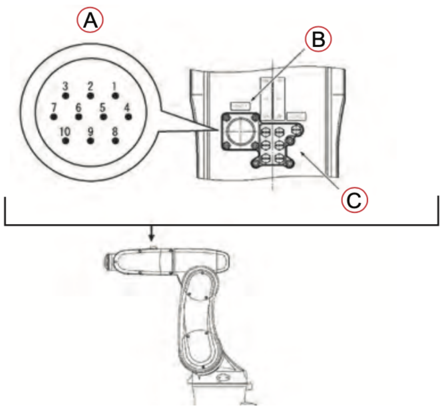


Figure 46: Viper pins [11]

6.7 Implementing the System in Real Life

To implement this solution in real life, several factors must be considered. At Westing, they stack two layers, sometimes adding a third layer on top with smaller buckets, and wrap plastic around to stabilize the stack. This process is very challenging for a robot. While a robot could stack a third layer, wrapping it could cause the buckets to fall.

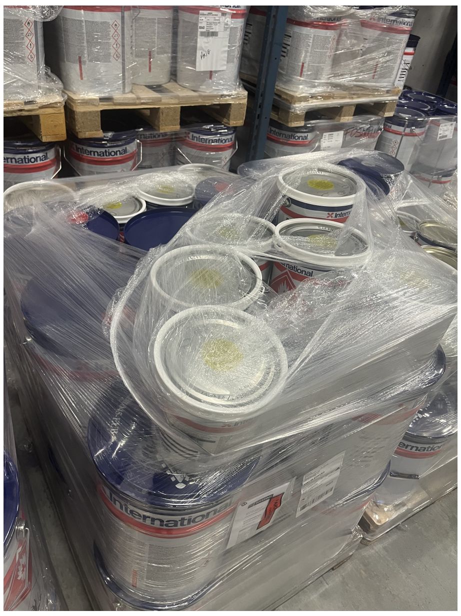


Figure 47: Third layer looks unstable

At Westing, this is not an issue because they prefer to automate the heavy lifting and manually handle the smaller buckets. However, this may not be the case for every warehouse.

Another challenge is that all buckets above 2 meters in the warehouse must be wrapped in plastic for safety. However, the robot requires the buckets to be unwrapped in order to pick them.

Some pallets in the warehouse have cardboard placed between the buckets. This cardboard must be removed for the robot to pick up the buckets.

Lastly, almost every product inspected at Westing was marked with class 3 flammable and class 8 corrosive labels. To address potential hazards, any robot operating in the area should have protection against short circuit ignition and resistance to corrosive substances in case of undetected spills.

6.8 Buckets

The paint buckets were donated to us by Enrico Agostinelli, Operating Technician at NTNU. The buckets had to be emptied for any residue and thoroughly cleaned before they were brought to the testing environment in Manulab.

Since the buckets had to be emptied, they weighed almost nothing. Testing with the actual weight of different paint buckets would have been more beneficial for the solution.

There were some size variations in the donated buckets. These size variations were beneficial to test the solution, as a size variation in a typical warehouse order is common. This ensured a more accurate simulation of a warehouse environment.

An identified weakness for the buckets stored in the database is that it consist of normalized paint buckets. All the paint buckets have uniform lids. It would have been beneficial to the testing if buckets with "anomalies" such as plugs in the lid were also used. The objective was to manage identifying and stacking paint buckets, but being able to performed testing on a varied selection of lids could have provided more data. This data could have helped identify any challenges or adaptations needed, if any. In figure 48, some of the testing buckets are shown.

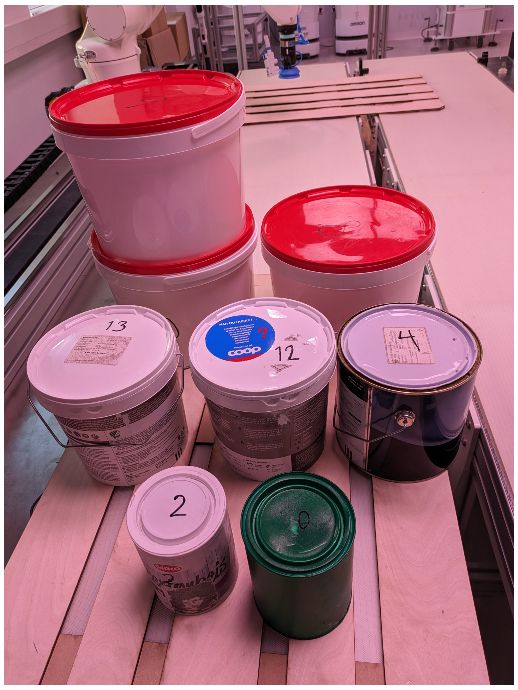


Figure 48: Typical buckets used in the testing

Custom end effector: The suction cup used in this project was a generic suction cup. By designing a custom end effector that is specifically designed for the task, the robot could be made more efficient and reliable. This could include a custom suction cup that is both segmented and compliant to better fit a wider range of buckets, as well as offer multiple points of contact for stability under movement as seen in figure 13.

7 Conclusion

This bachelor assignment was part of the assignments given by NTNU Ålesund, issued by Solwr. The main goal of the assignment was to develop a solution for picking and placing paint buckets.

Trough conducting research a solution was developed. This solution consists of using a Viper robot arm to move the paint buckets in cohesion with a Lidar camera. The camera detects the paint buckets corresponding to the buckets given as input.

To develop the solution, an algorithm handling order input and placements was developed in Python. To detect the paint buckets, an algorithm using Ransack was also developed. Programs handling the steering of the robots movements were written in ACE and Sysmac.

The resulting algorithms and programs were able to handle the detection and placement of the paint buckets, with some minor issues with placing on the pallet. The issue was assumed to be a malfunction with the Lidar camera, and upon borrowing the same type of camera from Solwr, it was concluded the deviation occurred with both cameras.

Bibliography

[1] Solwr. *A SOLWR SOLUTION Experience Grab.* URL: https://solwr.com/products/grab (visited on 8th Mar. 2024).

[2] *Intel® RealSense™ LiDAR Camera L515 — intelrealsense.com.* https://www.intelrealsense.com/lidar-camera-l515/. [Accessed 27-05-2024].

[3] Omron. *Automation Control Environment (ACE).* URL: https://www.ia.omron.com/products/family/3521/ (visited on 5th May 2024).

[4] Omron. *Sysmac Studio Software.* URL: https://automation.omron.com/en/us/products/family/sysstdio (visited on 12th Apr. 2024).

[5] ASKO. *ASKO investerer i teknologiselskapet Solwr.* URL: https://asko.no/nyhetsarkiv/asko-investerer-i-teknologiselskapet-solwr/ (visited on 8th Mar. 2024).

[6] Vital vision technology. *Zivid 2+ M130.* URL: https://vitalvisiontechnology.com/zivid-2-m130/ (visited on 22nd May 2024).

[7] Omron. *Viper Articulated robot for machining, assembly, and material handling.* URL: https://industrial.omron.eu/en/products/viper (visited on 8th Mar. 2024).

[8] Intel realsense. *Intel® RealSense™ LiDAR Camera L515.* URL: https://www.intelrealsense.com/lidar-camera-l515/ (visited on 22nd May 2024).

[9] NTNU. *Manulab – Idea Lab – NTNU Ålesund.* URL: https://www.ntnu.no/blogger/ihb/manulab/ (visited on 24th May 2024).

[10] Amatec Automasjon. *NTNU MANULAB - et visjonært prosjekt.* URL: https://www.amatecautomasjon.no/nyheter/industri-4-0-lab-til-ntnu (visited on 24th May 2024).

[11] Omron. *Viper 650/850 Robot with eMB-60R.* URL: https://assets.omron.eu/downloads/latest/manual/en/i599_viper_650,_850_robot_with_emb-60r_users_manual_en.pdf?v=5 (visited on 29th Apr. 2024).

[12] Omron. *Articulated Robots Viper 850.* URL: https://assets.omron.com/m/5fc8a2f359c9ed12/original/-Viper-850-Datasheet.pdf (visited on 8th Mar. 2024).

[13] Omron. *eV+ Language.* URL: https://edata.omron.com.au/eData/Robotics/I604-E-01.pdf (visited on 18th Apr. 2024).

[14] *What is OPC? - OPC Foundation — opcfoundation.org.* [Accessed 27-05-2024].

[15] *Classic - OPC Foundation — opcfoundation.org.* [Accessed 27-05-2024].

[16] *Unified Architecture - OPC Foundation — opcfoundation.org.* [Accessed 27-05-2024].

[17] Python Software Foundation. *Python, about.* URL: https://www.python.org/about/ (visited on 25th May 2024).

[18] Juraj Novák. *SQLite Viewer.* URL: https://inloop.github.io/sqlite-viewer/ (visited on 8th May 2024).

[19] The Matplotlib development team. *Matplotlib: Visualization with Python.* URL: https://matplotlib.org/ (visited on 22nd May 2024).

[20] NumPy team. *NumPy.* URL: https://numpy.org/ (visited on 27th May 2024).

[21] Martin A. Fischler and Robert C. Bolles. 'Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography'. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[22] Angel Manzur. *Got outliers? RANSAC them!* URL: https://medium.com/@angel.manzur/got-outliers-ransac-them-f12b6b5f606e (visited on 30th Apr. 2019).

[23] Julie Digne and Carlo de Franchis. 'The Bilateral Filter for Point Clouds'. In: *Image Processing On Line* 7 (2017), pp. 278–287. DOI: 10.5201/ipol.2017.179. URL: https://hal.science/hal-01636966.

[24] International computer science institute. *On-line Algorithms Versus Off-Line Algorithms: How Much is it Worth to Know the Future?* URL: https://www1.icsi.berkeley.edu/pubs/techreports/TR-92-044.pdf (visited on 22nd May 2024).

[25] John E. Shore. *On the External Storage Fragmentation Produced by First-Fit and Best-Fit Allocation Strategies.* URL: https://dl.acm.org/doi/pdf/10.1145/360933.360949 (visited on 23rd May 2024).

[26] Flávio K. Miyazawa. *Approximation Algorithms for Circle Packing.* URL: https://www.ime.usp.br/~spschool2016/wp-content/uploads/2016/07/Miyazawa.pdf (visited on 13th May 2024).

[27] a PTC Business Onshape. *Onshape — Product Development Platform — onshape.com.* https://www.onshape.com/en/. [Accessed 28-05-2024].

[28] Blender Foundation. *blender.org - Home of the Blender project - Free and Open 3D Creation Software — blender.org.* https://www.blender.org. [Accessed 28-05-2024].

[29] Lorenzo Neumann. *blainder-range-scanner.* https://github.com/ln-12/blainder-range-scanner. 2024.

[30] A. K. Jaiswal and B. Kumar. *VACUUM GRIPPER- AN IMPORTANT MATERIAL HANDLING TOOL.* URL: https://www.researchgate.net/profile/Binay-Kumar-8/publication/317823110_Vacuum_Cup_Grippers_for_Material_Handling_In_Industry/links/59f7fed60f7e9b553ebef487/Vacuum-Cup-Grippers-for-Material-Handling-In-Industry.pdf (visited on 17th May 2024).

[31] *CuPy — cupy.dev.* https://cupy.dev. [Accessed 28-05-2024].

Appendix

A Preliminary Submission

FORPROSJEKT - RAPPORT

FOR BACHELOROPPGAVE

NTNU
Kunnskap for en bedre verden

TITTEL: Solwr - Paint bucket picker

KANDIDATNUMMER(E):			
DATO: 17/01/24	EMNEKODE: AIS2900	EMNE: Bacheloroppgave ingeniørfag	DOKUMENT TILGANG: - Åpen
STUDIUM: AUTOMATISERING OG INTELLIGENTE SYSTEMER		ANT SIDER/VEDLEGG: 16/1	BIBL. NR: - Ikke i bruk -

OPPDRAGSGIVER(E)/VEILEDER(E): Olivier Roulet-Dubonnet (CTO Solwr robotics), Adam Leon Kleppe, Paul Steffen Kleppe

OPPGAVE/SAMMENDRAG:

Oppgaven går ut på å utvikle og demonstrere en prototype av et automatisert system som kan plukke malingsspann ved hjelp av en plukkerobot. Dette innebærer blant annet prototyping av en griper, testing av sugekopper som kan plukke malingsspann av forskjellig størrelse/vekt, og å bruke kunstig syn for identifisering av sylindriske objekt ved bruk av punktsky. Malingsspannene skal plasseres på en palle. En viktig del av oppgaven går ut på å kartlegge hvordan dette gjøres i dag.

Oppgaven er gitt av bedriften Solwr som er lokalisert i Ålesund.

Denne oppgaven er en eksamensbesvarelse utført av student(er) ved NTNU i Ålesund.

Preliminary Submission

NTNU I ÅLESUND Side 2
Forprosjektrapport – bacheloroppgave

INNHOLD

Preliminary Submission

1 INNLEDNING

Oppgaven ble valgt på bakgrunn av bachelorprosjekt ved NTNU Ålesund. Dette var gruppens foretrukne førstevalg da oppgaven er praktisk og vil gi relevant erfaring som bygger videre på kunnskapen studentene har tilegnet seg under studieforløpet. Oppgaven dekker også gruppemedlemmenes interesser. Oppdragsgiver er bedriften Solwr som holder til i Østre Gangstøvika 8, 6009 Ålesund.

Målet er å utvikle og demonstrere en prototype av et automatisert system for å plukke og plassere malingsspann.

Formålet med oppgaven er å utvikle og demonstrere en prototype av et automatisert system som kan plukke malingsspann ved hjelp av en robot. Dette innebærer blant annet prototyping av en griper, testing av sugekopper som kan plukke malingsspann av forskjellig størrelse/vekt, og å bruke kunstig syn for identifisering av sylindriske objekt ved bruk av punktsky. Malingsspannene skal plukkes og plasseres på en palle. En viktig del av oppgaven går ut på å kartlegge hvordan dette gjøres i dag. Arbeid med oppgaven kan involvere bruk av PLS, Zivid 2+ 3D kameraer, drivere og diverse utstyr som motorer, sugekopper og vakuum pumper.

Oppgaven er gitt på bakgrunn av at kunder ønsker en løsning til å gripe sylindriske objekter i tillegg til firkantede. Derav er oppgaven rettet inn mot å gjøre det mulig å plukke sylindriske malingsspann i flere størrelser.

Prototyping av løsningen vil bli gjort på manulab hos NTNU, ved bruk av en Viper 850 robotarm. Når dette er gjort, vil vi forsøke og implementere dette hos Solwr og tilpasse løsningen til deres system dersom tiden strekker til og det er behov for det.

2 BEGREPER

Begrep	Beskrivelse
3D printing	Digitale modeller kan fysisk printes av en 3D-printer
Griper	Et verktøy som gjør det mulig å gripe et objekt ved forflytning
Kunstig syn	Gjenkjenning av objekter ved bruk av datamaskin
Zivid 2+ 3D kamera	et kamera som brukes i maskinsyn til å lage en punktsky
Punktsky	samling av datapunkter i et tredimensjonalt koordinatsystem
Viper 850	Artikulert robotarm som kan bevege seg i 6 akser

3 PROSJEKTORGANISASJON

3.1 Prosjektgruppe

Studentnummer(e)
554931 – Marianne Nerland 498771 – Thomas Hellstrøm Olsen 538165 – Melina Nygard Karlsen

3.1.1 Oppgaver for prosjektgruppen - organisering

3.1.2 Oppgaver for prosjektleder

Prosjektleder: Melina Nygard Karlsen

Prosjektlederens ansvarsområde inkluderer å ha det overordnede ansvaret for hele prosjektets gjennomføring. Dette inkluderer en helhetlig oversikt og styring av alle prosessene involvert. Arbeidsoppgavene inkluderer:

- Planlegging og organisering av prosjektet
- Kommunikasjon mellom de forskjellige partene i prosjektet (teammedlemmer, ledelsen og andre relevante parter)
- Problemløsning (være forberedt på å takle problemer som oppstår underveis)
- Risikostyring (identifisere, vurdere og håndtere risikoer som kan påvirke prosjektet)
- Møteinnkalling
- Budsjettkontroll
- Sørge for at prosjektet holder seg innenfor relevante lover, forskrifter og bransjestandarder.

3.1.3 Oppgaver for sekretær

Sekretær: Thomas Hellstrøm Olsen

Ansvarsområde
- Føre logg fra møter og statusmøter/standup.
- Sørger for at gruppemedlemmer holder seg til avtalte protokoller

Arbeidsoppgaver
- Skriver referat fra alle møter med bedrift og veileder.
- Skriver referat fra interne møter / standup
- Sjekker at medlemmer har loggført timer
- Oppdaterer Gant diagram

3.1.4 Oppgaver for øvrige medlem(mer)

Øvrige medlem: Marianne Nerland

Ansvarsområder
- Delta i planlegging og organisering av prosjekt
- Bidra med oppgaver til prosjektleder og sekretær

Arbeidsoppgaver
- Jobbe med tildelte arbeidsområder

3.2 Styringsgruppe (veileder og kontaktperson oppdragsgiver)

Veiledere: Adam Leon Kleppe og Paul Steffen Kleppe

Oppdragsgiver: Olivier Roulet-Dubonnet (CTO Solwr robotics)

Kontaktperson: Rasmus Nes Tjørstad

4 AVTALER

4.1 Avtale med oppdragsgiver

I forbindelse med bachelorprosjektet, har vi planlagt å utføre tester med ulike sugekopper for å finne den mest effektive modellen for håndtering av malingsspann i varierende størrelser og vekter. Vi skal også bruke et Zivid 2+ kamera for å nøyaktig detektere sylindriske objekter i en punktsky. Vår tilnærming til oppgaven er fleksibel, og vi har muligheten til å utforske ulike metoder og teknikker for å nå våre mål.

Vi har inngått en avtale med oppdragsgiveren om å låne nødvendig utstyr, inkludert Zivid kameraet, og eventuelt annet tilleggsutstyr som sugekopper dersom det er behov for det. En essensiell del av prosjektet er å forstå og dokumentere hvordan dette gjøres i dag. I denne forbindelse har vi også avtalt å gjennomføre en grundig kartlegging for Solwr hos bedriften Westing i Fosnavåg. Dette vil gi oss verdifull informasjon som kan bidra til å forbedre vår egen tilnærming og effektivitet i prosjektet.

4.2 Arbeidssted og ressurser

Vi har tilgang på arbeidsplass hos Solwr, og også hos NTNU. NTNU kommer til å være vårt hovedarbeidssted (I hvert fall i begynnelsen).

Hos Solwr har har vi tilgang til de ressursene som vi trenger, som kamera, PLS, vakuum pumpe/sugekopper, 3d-printere, motorer og drivere. De kan også skaffe annet utstyr dersom det trengs. Hos NTNU har vi tilgang til Viper robotene, som vi i hovedsak skal bruke for å teste systemet vårt, og også diverse annet som kan være nyttig, slik som laser kutter, 3d-printere og diverse hardware.

I tillegg til dette, har vi tilgang til erfarne fagfolk, som kan bistå med feilsøking etc. dersom det blir behov for det.

4.3 Gruppenormer – samarbeidsregler – holdninger

Holdninger:

- **Alle medlemmer *SKAL* gjøre seg rede for regler i henhold til juks og bruk av KI**
- Selv om sannsynligheten tilsier en lavere karakter, sikter gruppen inn for karakteren «A» og mengden innsats dette krever
- Gruppen er innstilt på en jevn arbeidsflyt og skal være frempå med håndtering av forsinkelser.
- Medlemmer skal bidra til å holde godt arbeidsmiljø og gruppemoral.
- Gruppen skal holde fokus på korrekt kildebruk og sitering.
- Arbeid skal produseres med fornuft og bærekraft som en baktanke.
- Gruppen setter hovedfokus på å arbeide mest mulig på skole/bedrift.

Samarbeidsregler:

- For å hindre utbrenthet skal søndag og helst lørdag holdes arbeidsfritt med mindre en force majure inntreffer.
- Medlemmer skal føre timer for alt arbeid med bacheloroppgaven.
- Det skal føres rapport hver uke på fredag, enten som del av oppsummering med veileder/bedrift eller fra intern koordinering.
- Hver fredag avsluttes med et internt gruppemøte for gjennomgang av GANT diagram og status av prosjekt.
- Normal arbeidstid defineres som arbeidsdager 09:00 til 16:00. Dette blir kjernetid der gruppen skal jobbe sammen, eller lett tilgjengelig av hverandre.
- Mens INGA faget kjører blir kjernetid forbeholdt torsdag og fredag, og ellers ledig tid blir anvendt hensiktsmessig.
- Siden et gruppemedlem ikke har INGA faget skal medlemmet anse normal arbeidstid 09:00 til 16:00 hver ukedag.

5 PROSJEKTBESKRIVELSE

5.1 Problemstilling - målsetting - hensikt

Problemstilling: Hvordan kan Solwr utvikle en automatisert løsning med deres plukkerobot for å plukke malingsspann fra paller på en effektiv og pålitelig måte? Utfordringen ligger i å utforme en griper som kan tilpasse seg forskjellige størrelser på malingsspann, og integrere 3D kamera for identifisering og håndtering av objektene.

Målet er å utvikle og demonstrere en prototype av et automatisert system for å plukke malingsspann. Systemet skal være i stand til å identifisere og plukke opp malingsspann fra paller, og plassere dem strategisk på en annen palle. Vi har som mål å utvikle et effektivt system, så optimalisering vil være et fokus gjennom hele prosessen.

Prosjektet vil gi verdifull erfaring og kunnskap innen ulike automatiseringsteknologier som kunstig syn, sensorer, programmering og mekanisk design.

5.2 Krav til løsning eller prosjektresultat – spesifikasjon

- Kartlegge behov for plukking og stabling av sylindriske objekter.
- Bruke 3D og 2D kamera for å detektere og verifisere riktig objekt.
- Effektiv stabling av forskjellige størrelser malingsspann på en palle.
- Potensielt utvikle en griper.
- Potensiell griper skal forsøke å designes til å ha lang levetid, med tanke på økonomisk lønnsomhet.
- Den endelige løsningen skal klare å identifisere malingsspann samt unngå feilidentifisering med lav margin.
- Prosjektet skal fullføres innenfor økonomiske rammer som er realistiske/typiske for en bacheloroppgave.

5.3 Planlagt framgangsmåte(r) for utviklingsarbeidet – metode(r)

Vi har valgt å gå for Scrum som vår primære prosjektstyrings- og utviklingsmetode. Scrum er en bredt anerkjent og velprøvd metode innenfor smidig utvikling, og er spesielt effektiv for prosjektet vårt som innebærer rask utvikling og hyppige endringer.

Scrum er en iterativ og inkrementell tilnærming som legger vekt på teamarbeid, fleksibilitet og kontinuerlig forbedring. Denne metoden deler prosjektet inn i korte arbeidssykluser (kjent som sprinter), som typisk varer fra 2-4 uker. Hver sprint inneholder planlegging, utførelse, gjennomgang og refleksjon. Dette gjør det mulig for gruppen å raskt tilpasse seg endringer og levere delprodukter kontinuerlig.

Denne metoden er kjent for å fremme rask og fleksibel respons på endringer, noe som er avgjørende i et dynamisk utviklingsmiljø. Det forbedrer også kommunikasjonen og samarbeidet i gruppen, og sikrer en jevnlig og klar prosjektprogresjon.

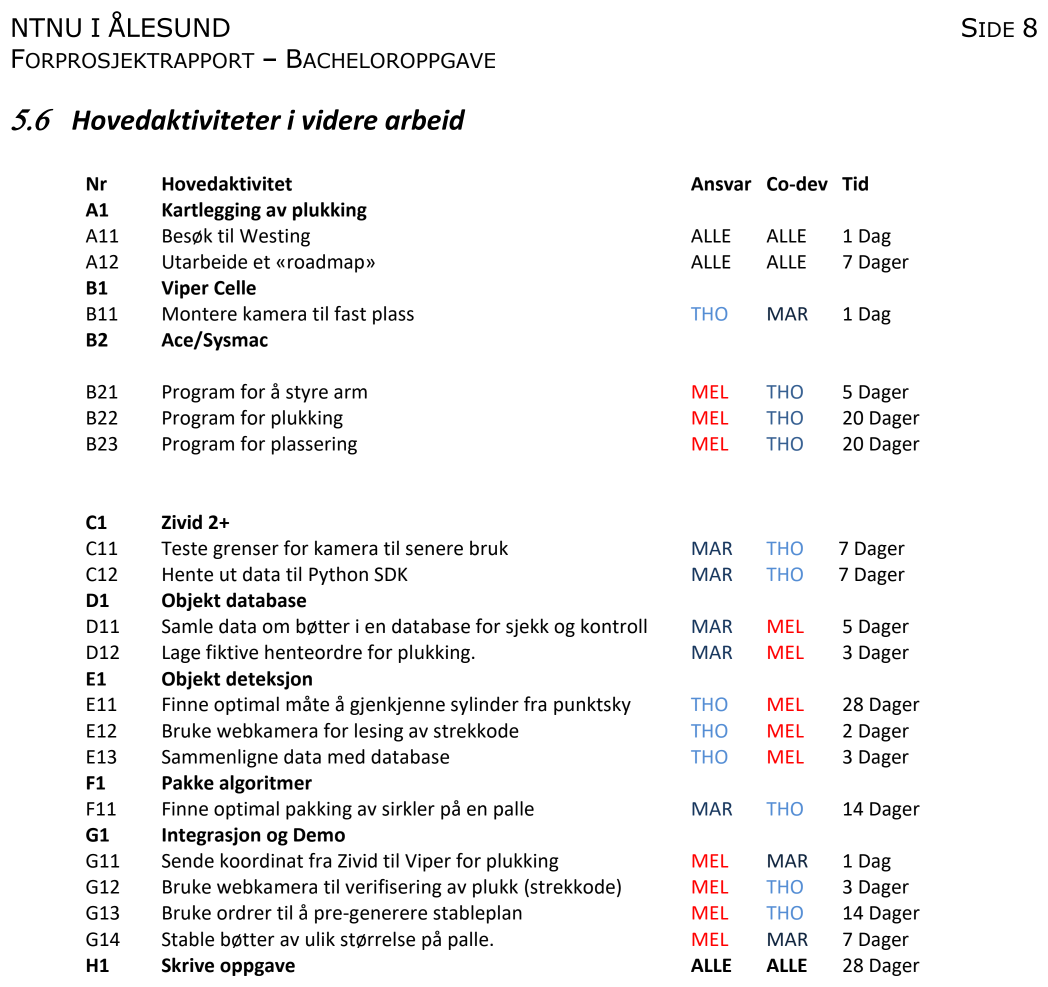
En kjent utfordring med Scrum er dens avhengighet av teamets selvorganisering og disiplin. Uten en klar forståelse og dedikasjon til Scrum-prinsippene, kan prosjektet møte forsinkelser og effektivitetsproblemer. For å overvinne disse mulige svakhetene, vil vi sørge for at alle gruppemedlemmene får en grundig gjennomgang av Scrum-prinsipper. Vi vil også utpeke en på gruppa som er ansvarlig for å veilede gruppen gjennom prosessen, og sikre at praksisene blir fulgt effektivt. Videre vil regelmessige sprintgjennomganger og retroperspektiver bli brukt for kontinuerlig å evaluere og forbedre vår tilnærming.

5.4 Informasjonsinnsamling – utført og planlagt

- Har hentet inn relevante datablader som vi har lagt til på gruppens wiki.
- Vi har avtale med manulab om å kunne benytte Vipercellen i manulab under bachelorprosjektet.
- Vi har begynt med opplæring til bruk av robot, og begynt å hente inn informasjon om roboten.
- Vi har begynt å lese om punktsky og skal hente inn informasjon hvordan dette skal benyttes i oppgaven.
- Denne uken skal vi uføre en kartlegging hos bedriften Westing i Fosnavåg.

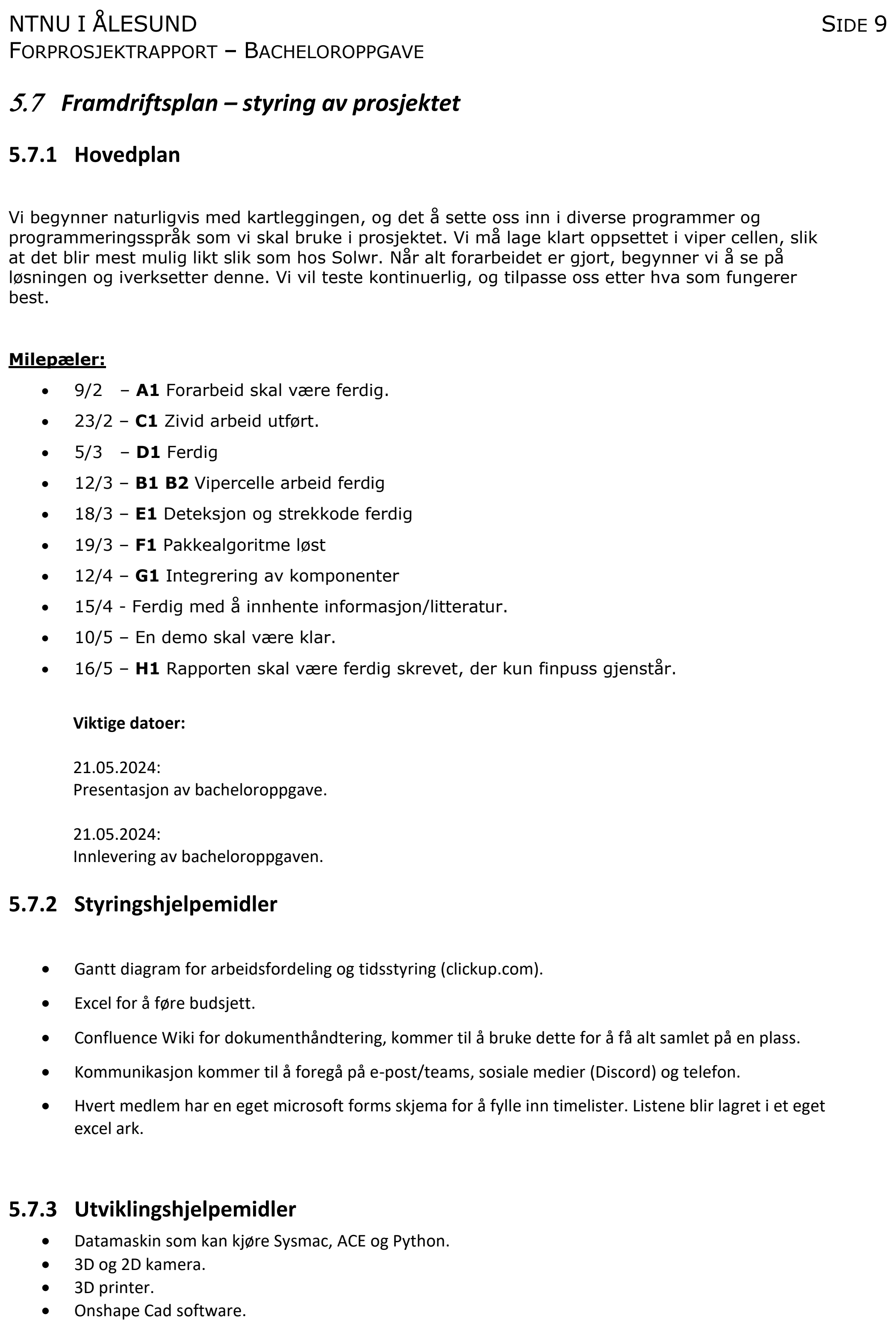
5.5 Vurdering – analyse av risiko

Element	Trussel	Sårbarhet	Tiltak	Risiko	Sannsynlighe t
Viper celle	Samkjøring og kontroll	Samkjøre ACE og SYSMAC	Starte tidlig med testing/ spør manulab ingeniører	Høy	Middels
Zivid 2+	Ikke klare å bruke data	Nytt kamera og lite brukt SDK	Starte tidlig med å research og testing av kamera	Middels	Lav
Deteksjons algoritmer	Forstå/ Iverksette	Ingen I gruppa har erfaring med feltet	Planlegge research for deteksjon, spørre veileder	Middels	Middels
Pakkealgoritmer	Forstå/ Iverksette	Ingen I gruppa har erfaring med feltet	Utforske relevante pakkealgoritme r	Middels	Middels
Designe griper for Ulike størrelser	Finne godt design uten å bruke for mye tid	3D printer er kanskje ikke tett nokk for vakum deler	Bytte til en eksisterende industriell løsning	Lav	Lav
Integrering av teknologier	Få alle underkomponente r til å fungere sammen	Flere forskjellige programvare r og teknologier	Starte tidligst mulig for å løse problemer.	Høy	Middels



5.6 Hovedaktiviteter i videre arbeid

Nr	**Hovedaktivitet**	**Ansvar**	**Co-dev**	**Tid**
A1	**Kartlegging av plukking**			
A11	Besøk til Westing	ALLE	ALLE	1 Dag
A12	Utarbeide et «roadmap»	ALLE	ALLE	7 Dager
B1	**Viper Celle**			
B11	Montere kamera til fast plass	THO	MAR	1 Dag
B2	**Ace/Sysmac**			
B21	Program for å styre arm	MEL	THO	5 Dager
B22	Program for plukking	MEL	THO	20 Dager
B23	Program for plassering	MEL	THO	20 Dager
C1	**Zivid 2+**			
C11	Teste grenser for kamera til senere bruk	MAR	THO	7 Dager
C12	Hente ut data til Python SDK	MAR	THO	7 Dager
D1	**Objekt database**			
D11	Samle data om bøtter i en database for sjekk og kontroll	MAR	MEL	5 Dager
D12	Lage fiktive henteordre for plukking.	MAR	MEL	3 Dager
E1	**Objekt deteksjon**			
E11	Finne optimal måte å gjenkjenne sylinder fra punktsky	THO	MEL	28 Dager
E12	Bruke webkamera for lesing av strekkode	THO	MEL	2 Dager
E13	Sammenligne data med database	THO	MEL	3 Dager
F1	**Pakke algoritmer**			
F11	Finne optimal pakking av sirkler på en palle	MAR	THO	14 Dager
G1	**Integrasjon og Demo**			
G11	Sende koordinat fra Zivid til Viper for plukking	MEL	MAR	1 Dag
G12	Bruke webkamera til verifisering av plukk (strekkode)	MEL	THO	3 Dager
G13	Bruke ordrer til å pre-generere stableplan	MEL	THO	14 Dager
G14	Stable bøtter av ulik størrelse på palle.	MEL	MAR	7 Dager
H1	**Skrive oppgave**	**ALLE**	**ALLE**	28 Dager



NTNU I ÅLESUND SIDE 9
FORPROSJEKTRAPPORT – BACHELOROPPGAVE

5.7 Framdriftsplan – styring av prosjektet

5.7.1 Hovedplan

Vi begynner naturligvis med kartleggingen, og det å sette oss inn i diverse programmer og programmeringsspråk som vi skal bruke i prosjektet. Vi må lage klart oppsettet i viper cellen, slik at det blir mest mulig likt slik som hos Solwr. Når alt forarbeidet er gjort, begynner vi å se på løsningen og iverksetter denne. Vi vil teste kontinuerlig, og tilpasse oss etter hva som fungerer best.

Milepæler:

- 9/2 – **A1** Forarbeid skal være ferdig.
- 23/2 – **C1** Zivid arbeid utført.
- 5/3 – **D1** Ferdig
- 12/3 – **B1 B2** Vipercelle arbeid ferdig
- 18/3 – **E1** Deteksjon og strekkode ferdig
- 19/3 – **F1** Pakkealgoritme løst
- 12/4 – **G1** Integrering av komponenter
- 15/4 - Ferdig med å innhente informasjon/litteratur.
- 10/5 – En demo skal være klar.
- 16/5 – **H1** Rapporten skal være ferdig skrevet, der kun finpuss gjenstår.

Viktige datoer:

21.05.2024:
Presentasjon av bacheloroppgave.

21.05.2024:
Innlevering av bacheloroppgaven.

5.7.2 Styringshjelpemidler

- Gantt diagram for arbeidsfordeling og tidsstyring (clickup.com).
- Excel for å føre budsjett.
- Confluence Wiki for dokumenthåndtering, kommer til å bruke dette for å få alt samlet på en plass.
- Kommunikasjon kommer til å foregå på e-post/teams, sosiale medier (Discord) og telefon.
- Hvert medlem har en eget microsoft forms skjema for å fylle inn timelister. Listene blir lagret i et eget excel ark.

5.7.3 Utviklingshjelpemidler

- Datamaskin som kan kjøre Sysmac, ACE og Python.
- 3D og 2D kamera.
- 3D printer.
- Onshape Cad software.

5.7.4 Intern kontroll – evaluering

Vi skal ha ukentlige statusmøter. Dersom noe trenger oppfølging, blir dette tatt med en gang. Vi skal så godt det lar seg gjøre å ordne opp i ting selv, dersom dette ikke er mulig, kontakter vi veileder/kontaktperson for videre oppfølging.

Et mål/delmål er nådd når vi kan bruke produktet i neste steg uten problemer.

5.8 Beslutninger – beslutningsprosess

Beslutninger om avgrensing/presisering av oppgaven og andre sentrale beslutninger ble tatt av styringsgruppen sammen med oss. Dette ble gjort under ett møte som vi hadde 19/1.

Videre, vil små beslutninger som ikke endrer oppgaven diskuteres/bestemmes i gruppen, og store viktige beslutninger diskuteres med veileder og bedrift.

6 DOKUMENTASJON

6.1 Rapporter og tekniske dokumenter

- Møtereferat fra alle møter
- Timelister
- Vi skal også lage en grundig dokumentasjon av:

- Plukkingen
- Plasseringen
- Teorien bak algoritmene vi bruker
- Griper/sugekopper

7 PLANLAGTE MØTER OG RAPPORTER

7.1 Møter

7.1.1 Møter med styringsgruppen

Vi har ikke satt opp noen fast dato til møter med styringsgruppen. Dette tenker vi at vi tar fortløpende dersom det er behov for det. Vi skal ha møte med veileder en gang i uken. Dette vil skje på fredager klokken 10.00. Det vil bli skrevet møtereferat.

7.1.2 Prosjektmøter

Hver fredag før møtet med veileder skal vi ha et internt statusmøte for kartlegging og eventuell justering av arbeid. Dette skal loggføres som hvilket som helst annet møte.

Preliminary Submission

NTNU I ÅLESUND SIDE 11
FORPROSJEKTRAPPORT – BACHELOROPPGAVE

7.2 Periodiske rapporter

7.2.1 Framdriftsrapporter (inkl. milepæl)

Det er planlagt statusrapporter hver uke (fredag) sammen med veileder. Disse rapportene vil inkludere eventuelle milepæler. Oppdragsgiver vil også få tilgang til disse om ønskelig.

Februar
- **24.02.02:** Ukentlig rapport
- **24.02.09: A1** Forarbeid skal være ferdig
- **24.02.09:** Ukentlig rapport
- **24.02.16:** Ukentlig rapport
- **24.02.23:** Ukentlig rapport
- **24.02.23: C1** Zivid arbeid utført

Mars
- **24.03.01:** Ukentlig rapport
- **24.03.05: D1** Objekt database ferdig
- **24.03.08:** Ukentlig rapport
- **24.03.12: B1 B2** Vipercelle ferdig
- **24.03.15:** Ukentlig rapport
- **24.03.18: E1** Deteksjon og strekkode ferdig
- **24.03.19: F1** Pakke algoritme ferdig
- **24.03.22:** Ukentlig rapport
- **24.03.29:** Ukentlig rapport

April
- **24.04.05:** Ukentlig rapport
- **24.04.12:** Ukentlig rapport
- **24.04.12: G1** Integrering av milepæler.
- **24.04.15:** Innhenting av informasjon/litteratur ferdig.
- **24.04.19:** Ukentlig rapport
- **24.04.26:** Ukentlig rapport
- **24.04.29:** Fullført innhenting av informasjon/litteratur

Mai
- **24.05.03:** Ukentlig rapport
- **24.05.10:** En demo skal være klar
- **24.05.10:** Ukentlig rapport
- **24.05.16:** Rapporten skal være ferdig skrevet, der kun finpuss gjenstår
- **24.05.16:** Ukentlig rapport
- **24.05.19:** Ferdigstille bachelorpresentasjon
- **24.05.20:** Finpusse frem til Innlevering
- **24.05.21:** Presentasjon bacheloroppgave
- **24.05.21:** Innlevering av bacheloroppgaven

8 PLANLAGT AVVIKSBEHANDLING

Hva skal gjøres dersom prosjektet (framdrift/innhold) ikke går som planlagt:

- Gruppen må samarbeide om å identifisere avvik, og loggføre disse slik at det holdes en oversikt over hva som ikke går etter opprinnelig plan samt konsekvensene.
- Gruppen må sammen vurdere alvorlighetsgrad av avvik samt hvilke tiltak som kan implementeres der det lar seg gjøre.
- Det kan avtales møte med veileder om avvik/tiltak som vurderes som å ha betydelige konsekvenser for planen.

Planlagt prosedyre for endringer:

- Dokumentere endringer i planen der det oppstår behov for endring. Disse endringene skal gjøres ved enighet i gruppen.
- Gruppen vil informere bedrift og veileder ved behov basert på hvilke endringer som er gjort.
- Endringer vurderes opp mot budsjett og tidsramme der det er relevant, der endringer som vil ha betydelig innvirkning her konfereres med veileder og bedrift.

Ansvar:

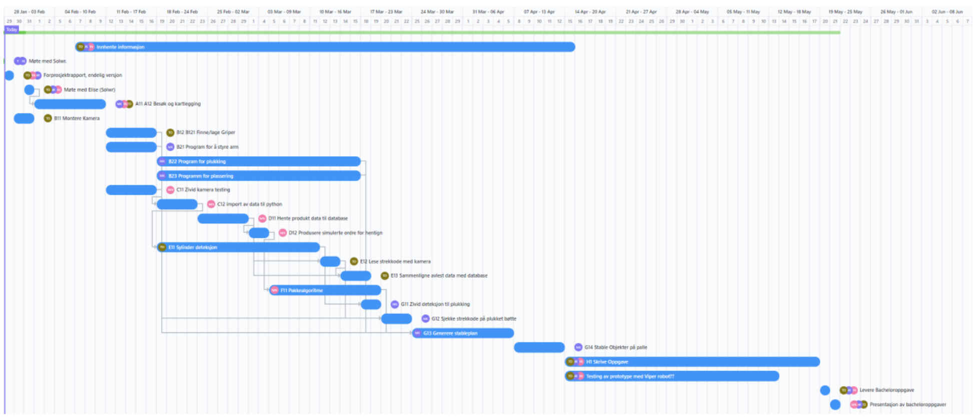
- Hvert gruppemedlem er ansvarlig for dokumentering av avvik og informere resten av gruppen samt bedrift/veiledere ved behov.

9 UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING

- Viper celle hos manulab
- Zivid 2+ kamera
- PC å kjøre kode / Ace / Sysmac

10 REFERANSER

VEDLEGG



B Gantt Diagram

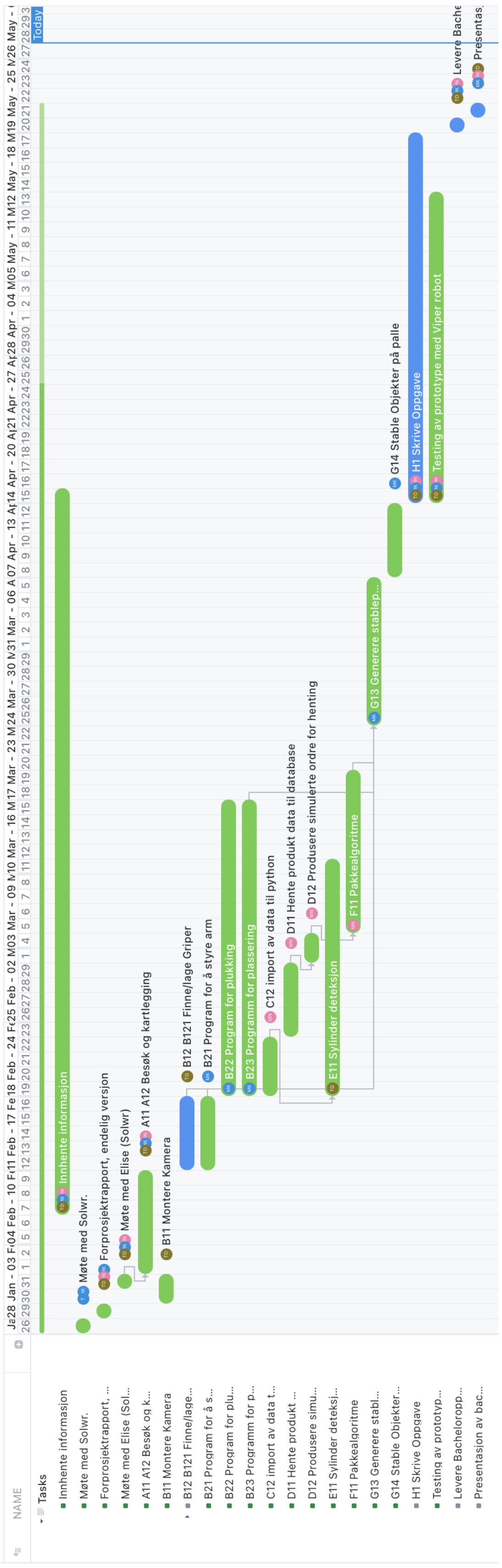


Figure 49: Gantt Diagram