Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Edvard Rauø Vasdal

Mixed box palletizing; a suggestion to detect and pack unknown sized objects in bins using point clouds and genetic algorithms

Master's thesis in Cybernetics and Robotics
Supervisor: Annette Stahl
June 2019

NTNU
Norwegian University of Science and Technology

Edvard Rauø Vasdal

Mixed box palletizing; a suggestion to detect and pack unknown sized objects in bins using point clouds and genetic algorithms

Master's thesis in Cybernetics and Robotics
Supervisor: Annette Stahl
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Preface

This thesis concludes my Master of Science degree in Cybernetics and Robotics at Norges Teknisk-Naturvitenskapelige Universitet (NTNU). The work has been carried out during the spring of 2019, and the study has been completed in collaboration with Rocketfarm AS, which is a company that works with solutions that include industrial robots.

I was introduced to 3D-data for the first time in the specialization project. Ahead of the master thesis, I had no previous knowledge about Bin packing problems and their solutions. For these reasons, there have been many hours used to understand the problems and in self-education. All the software for the project was developed by myself with some use of third-party libraries. For the Box detection, there is used Point Cloud Library, PCL, and the tools it provides. For the Robotic bin packing, there is only used Numpy. The equipment that was provided to me for this master thesis was: an office computer and a lab spot with a Zivid-camera.

I want to thank my supervisor Annette Stahl for academic supervision and having a guidance meeting with me every other week. Furthermore, I would like to thank my primary contact in Rocketfarm AS, Lars Bårdgard Åstveit, that gave me the opportunities to work with them and these problems. Also, for being a great support and share useful knowledge about the industrial experience. Lastly, I will give a big thanks to my colleagues in the office, to Øystein Brox and Christian Aschehoug. There have been many good professional conversations, and not least for their contribution to a good social environment.

Trondheim June 16, 2019
Edvard Rauø Vasdal

Abstract

Palletizing of boxes has been done for robots for many years. It is a simple task to pick a box and place it at a pallet. However, if the box has unknown size and positions, the solution to pick-and-place is not trivial. In this thesis, there are going to be addressed two methods that are meant to be a part of a fully automated palletizing system. The first method detects the sizes and poses of all boxes that are present in a point cloud. The second method decides which box should be picked first and where to place it.

The Box detection method is developed by the use of common point cloud operations and self-developed methods. All of these are implemented in C++ with the help of point cloud library, PCL. In order to map the result of all the parts of the method, there is developed a point cloud simulator. The box detection method is tested on point cloud from the simulator and Zivid-camera. The results show accuracy and precision that is far greater then what industrial applications need with a computational-time approximately equal to the capture time of the Zivid-camera.

The second method is the Robotic bin packing. By utilizing the research done on Bin packing problem, the method calculates the future cost of the available boxes and current solution. With this cost, the decision of which box to pick first is made. This is implemented in Python. The method is tested on data randomly generated with industrial properties. The performance of the methods show significant improvement compared to a simple heuristic packing, and in many cases optimal.

The results of the two methods show that both of the methods could be implemented in an industrial application. By implementing the findings in this thesis in industry, the shipping and distribution sector may obtain a significant step towards a more optimal process. The consequences for such an optimization would be reduced need for freight carriers, thus lowering the cost for shipping and reducing the carbon footprint in this sector.

Sammendrag

Palletering av bokser er gjort med roboter i mange år. Det er en enkel oppgave å velge en boks og stable den på en pall. Men hvis boksen har ukjent størrelse og posisjoner, er løsningen til dette problemet ikke triviell. I denne oppgaven skal to metoder som er ment å være en del av et fullt automatisert palleteringssystem utvikles. Den første metoden oppdager størrelser, posision og orientering av alle bokser som er tilstede i en scene. Den andre metoden bestemmer hvilken boks som skal velges først og hvor den skal plasseres.

Boksdeteksjonsmetoden er utviklet ved bruk av vanlige punktsky-operasjoner og selvutviklede metoder. Alle disse er implementert i C ++ ved hjelp av point cloud library, PCL. En punktskyssimulator er utviklet for å kartlegge resultatene av alle delene av metoden. Boksdeteksjonsmetoden er testet på punktskyer fra simulatoren og Zivid-kameraet. Resultatene viser nøyaktighet og presisjon som er større enn hva industrielle applikasjoner krever, og med en beregnings-tid omtrent som opptakstiden til Zivid-kameraet, er ytelsen tilstrekkelig.

Den andre metoden er Robotic bin packing. Ved å utnytte forskningen som er gjort på Bin packing problem, metoden regner ut fremtidige kostnader for de tilgjengelige basert på fremtidige bokser og nåværende løsning. Med denne kostnaden er beslutningen om hvilken boks å velge først gjort. Alt dette er implementere med bruk av Python. Metoden er testet på data tilfeldig generert med typiske industrielle egenskaper. Utførelsen av denne metoden viser betydelig forbedring sammenlignet med en ren erfaringsbasert pakking, og i mange tilfeller optimal pakking.

Resultatene fra de to metodene viser at begge metodene kunne implementeres i en industriell applikasjon. Ved å implementere funnene i denne oppgaven i industrien, kan frakt- og distribusjonssektoren få et betydelig skritt mot en mer optimal prosess. Konsekvensene for en slik optimalisering vil være redusert behov for fraktbåter, og dermed redusere kostnadene for frakt og redusere karbonavtrykket i denne sektoren.

Contents

List of Tables

List of Figures

Abbreviations

EMS	=	Empty Maximal Space
ESMS	=	Empty Supported Maximal Space
BPP	=	Bin packing problem
GA	=	Genetic algorithm
PCL	=	Point Cloud Library
RANSAC	=	Random sample consensus
RBPP	=	Robotic bin packing problem

Chapter 1

Introduction

1.1 Motivation

Robots have been used in many years for the palletizing task. In a manufacturing facility, there are usually a homogeneous set of boxes, i.e., equally sized, which makes the picking and palletizing highly repetitive. In the case of a logistics facility, the bins can be a strongly heterogeneous set of boxes, i.e., varying in size. That makes it non-trivial for the robot how to pick and palletizing the boxes. Therefore this task is often done by manual labor. With the increasing number of boxes transported we can see that there is a demand for more solutions that utilize a robot for packing.

With the development of digitalization and automation during the last years, there has been much improvement in the field of 3D vision. After Microsoft announced the Kinect, in 2009, both the consumer market and the professional marked got an affordable camera. That led to software like Point Cloud Libary, PCL, in 2010, which made algorithms for 3D point cloud available. In 2017 the first structured light camera from Zivid was delivered to the marked. The Zivid-One camera has much higher accuracy than the Microsoft Kinect and is easy to use. This makes it ideal for solving the problem of detecting boxes for a robotic manipulator.

To address the problem of palletizing the boxes, the study of Bin packing problems, BPP, is of big interest. This is a widely studied problem in combinatorial optimization, and there are various formulations to different types of BPP, more described in section 2.7.2. The common challenge in BPP is grouping a set of small objects into one or several larger objects, for instance boxes on a pallet in the most optimal way. The general approach to solving such problems is based on changing the order the smaller objects are placed into the larger object. In many industrial packing cases, we only know which boxes that will be placed on each pallet, not the order they will arrive and could choose between a few numbers of boxes to pack next, which is the case of this thesis.

1.2 Problem description

Rocketfarm AS in Sogndal is delivering robotic solutions for industrial automation. For the last years, they have developed software for palletizing in manufacturing facilities. To evolve into more cases, they want to investigate how they can proceed towards palletizing with varying box sizes. Two of the most challenge part of this is to detect the boxes that are present and find a suitable placement for a box on the pallet.

The formal description of these two problems are stated as follows:

Problem statement

1. Given a point cloud, develop a method that will detect multiple boxes within the point cloud. Then for each of the boxes, the method should estimate the pose, position and orientation, and the size.
2. Develop a method that will estimate a solution for a Robotic bin packing problem, see Definition 1, page 22.

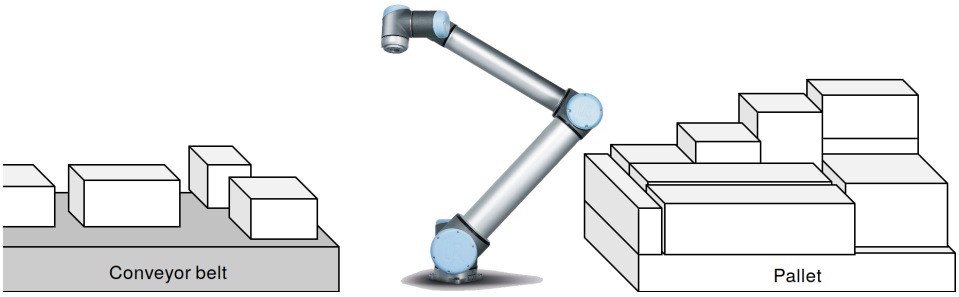


Figure 1.1: For a robotic manipulator to pick a box, it needs to know the pose and size, problem 1. To pack a pallet it needs to know which box to pick and where to place it, problem 2.

1.2.1 Limitation

In order to make these problems feasible for a Master thesis, it was necessary to introduce some limitations to the problem.

The limitations for the Box detection is as follows:

- There is no physical contact between the boxes that will be picked.
- The detection of boxes should be done within a second.
- The environment is stationary when a point cloud is captured.
- The surfaces of the boxes are possible to capture with a structured light camera.

And for the Robotic bin packing:

- The sizes of the boxes that should be known, but not the order.
- The boxes will have 3 different footprints and 3 different heights, i.e., in total 9 different boxes types.
- Rotations of the boxes is not to be considered.
- There are 3 boxes available for picking at each time.

1.3 Contribution

- **Box detection:** In a fully automated palletizing system, the boxes that are present are unknown. To solve this, there have been developed and implemented a solution to detect the size and pose of the boxes in the scene. This software is made in C++ by using the point cloud library, PCL. It has been tested both on data from the Zivid-camera and data from the simulator. The method has high accuracy and precision, more than sufficient for an industrial application. In addition, the computational time is almost the same, few milliseconds slower, as the capture time of the Zivid-camera.
- **Point cloud simulator:** To avoid manual and inaccurate measurements on testing data, there has been developed a point cloud simulator. With this simulator, it is possible to place a point cloud generated from CAD models in the scene. All geometric properties, pose and size, are known for all object, and the pose of the camera can be changed. There is also a possibility to output the mask for each point present in the scene, which makes it possible to train and test segmentation neural networks.
- **Bin packing:** In order to solve the Robotic bin packing problem, there had to be made a solution to the Bin Packing problem. To solve this, there has been implemented a Generic algorithm with a heuristic packing strategy inspired by research in the field. The performance is tested and it is sufficient for using it in the Robotic bin packing method.
- **Robotic bin packing:** Since the Robotic bin packing problem is not a well-studied, a new design had to be made. It is based on computing the best future cost for the available boxes and choose the box with the best cost. In order to make the solution that is achievable for a robotic manipulator, a new concept for manage feasible packing space have been developed. It guarantees that the packable space has support either by floor or a box underneath. The whole Robotic bin packing method is implemented in Python and tested on industrial relevant data. The performance of the method is equal or better compared to a pure heuristic packing method, and in many cases optimal.
- **Instance generator:** In lack of suitable datasets that fit the limitations for this thesis, there has been developed an instance generator. It generates a random instance that is industrially relevant and ensures that there exist a solution to the problem if it is solved as a Bin packing problem.

1.4 Structure of this report

This report includes:

- **Chapter 1**, presents the motivation for the problem and the problem that is worked on in this thesis. Also, the limitations to the problem and contribution are presented.
- **Chapter 2**, presents the tools used or considered used in box detection. To get sufficient knowledge about the Bin packing problem, there will be given an overview. It includes a definition of the general problem, typical variations of the problem, a literature review of different solutions, and lastly a definition of the Robotic bin packing problem.
- **Chapter 3**, describes the different parts of the two problem investigated, namely Box detection and Robotic bin packaging. For the Box detection, it consists of a point cloud preprocessing, the detection, and feature extraction. For the Robotic bin packing it consists of developing the heuristic packing strategy, a Generic algorithm to solve the Bin packing problem, and lastly use both of them in the Robotic bin packing method.
- **Chapter 4**, describes the experiments used to test the performance of the methods. The experiments include both synthetic data, and for the box detection real-world data. The performance of the method through the experiments are mapped and results are presented and discussed.
- **Chapter 5**, the conclusion of the thesis, and some suggestions to further work are presented.

Chapter 2

Background theory

This chapter is divided into the two main topics of this thesis, namely the Box detection and Robotic bin packing. For the first sections, there will be presented tools that are used as a part of the Box detection or have been considered to be a part of the Box detection. In the last section there will be build an in-depth knowledge about the Bin pack packing problem. This theory will later be used to develop a method for solving the Robotic bin packing problem.

2.1 Zivid-camera

The Zivid-One is the first 3D-calibrated camera that Zivid launched in 2017. To obtain an accurate point cloud, Zivid-One uses structured-light with 13 projected pattern to obtain one point cloud, and it takes 100 ms, 10 Hz. The minimum working depth is 600 mm, and the maximum working depth is 1100 mm. These are the optimal ranges and will give full overlap between the field-of-view of the projector and the image sensor. The output resolution of the point cloud is 1920×1200, and at a depth of 600 mm the image sensor will cover an area of 430×270 mm and at a depth 1100 mm the area is 780×490 mm.

The output of the Zivid-One is an organized point cloud. With that means that the points in the cloud can be indexed with rows and columns. This is convenient since then it is known that the neighboring pixel is the neighboring point, which can make searching in point cloud much faster. For each pixel in the Zivid point cloud, there is a value for a 3D position (XYZ), color (RGB), and a contrast value.

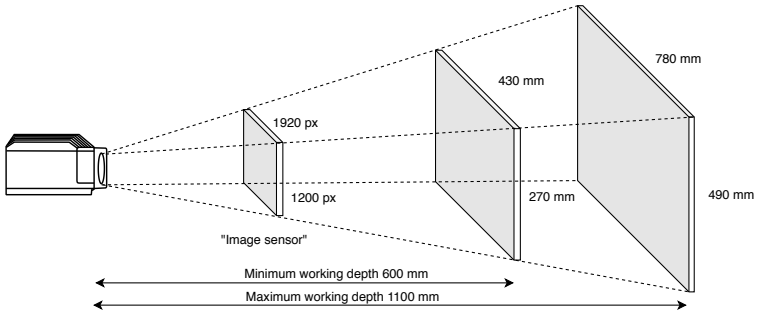


Figure 2.1: Working depth of a Zivid-camera.

2.2 Homogeneous coordinates

In order to describe the geometry in this thesis, there have been used Homogeneous coordinates. This mainly because it makes many of the operations done simpler than the same operations in the cartesian representation. Most of this thesis work with 3D geometric and will, therefore, begin with present the properties that have been used. For the case of 2D, many properties are valid from the 3D case. However, some important properties will be presented. The theory behind this is from (Hartley and Zisserman, 2003, p. 26-29, p. 66-68).

2.2.1 Homogeneous coordinates in 3D

A point in the 3D projective space, $\mathbb{P}^3$, is represented as:

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^\top \tag{2.1}$$

the equivalent to the representation in cartisian space is:

$$\mathbf{p} = \frac{1}{x_4} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3 \tag{2.2}$$

where x, y and z are the coordinates to the point. It can easly be seen that the $\mathbf{x} = \mathbf{p}$ when $x_4 = 1$.

In homogeneous coordinates for a point, $\mathbf{x}$, to be on a plane it has to satisfies the equation:

$$\mathbf{x}^\top \cdot \boldsymbol{\pi} = 0 \tag{2.3}$$

where $\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3 \pi_4)$. In inhomogeneous form of a plane is given by $\mathbf{p}^\top \mathbf{n} + d = 0$. Where $\mathbf{n}$ is normal vector, and the distance from origo is given by $d/||\mathbf{n}||$. For homogenogus form the first three components of $\boldsymbol{\pi}$ represents the the normal vector $\mathbf{n} = (\pi_1, \pi_2, \pi_3)$ and the distance therm $d = \pi_4$.

Given N points that are on the plane, then the plane, $\boldsymbol{\pi}$, is given by:

$$\begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_N^\top \end{bmatrix} \cdot \boldsymbol{\pi} = \mathbf{0}_{N\times 1} \tag{2.4}$$

If $N = 3$ and the points are linearly independent there can be seen that the points matrix has rank 3. Then the plane, $\boldsymbol{\pi}$, is given uniquely by the right null-space of the points matrix. For $N = 2$ the points are underdetermined and will have infinet amout of solutions to the problem. If $N > 3$ the points are overdetermined and there will be an approximated solution, e.g. with a leat-squars solution.

The intersection point, $\mathbf{x}$, between three planes, $\boldsymbol{\pi}_i$, can be computed easily by computing the right null-space of the 3×4 matrix composed by the planes:

$$\begin{bmatrix} \boldsymbol{\pi}_1^\top \\ \boldsymbol{\pi}_2^\top \\ \boldsymbol{\pi}_3^\top \end{bmatrix} \mathbf{x} = 0 \tag{2.5}$$

2.2.2 Homogeneous coordinates in 2D

Points and lines in 2D homogeneous coordinates are a reduction in one dimension from the 3D homogeneous coordinates. However, some elegant properties will be pointed out.

For a point, $\mathbf{x}$, to be on a line it has to satisfy:

$$\mathbf{l}^\top \mathbf{x} = 0 \tag{2.6}$$

Which means that the point $\mathbf{x}$ is orthogonal to the line $\mathbf{l}$.

By further investigating the orthogonality of a line and points, some interesting results will rise. If point $\mathbf{x}$ lies on both $\mathbf{l}_1$ and $\mathbf{l}_2$, then it means that $\mathbf{x}$ is the intersection between. It also means that $\mathbf{x}$ is orthogonal to both $\mathbf{l}_1$ and $\mathbf{l}_2$, that gives:

$$\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2 \tag{2.7}$$

An opposite argument can be maid by if a line $\mathbf{l}$ goes through both $\mathbf{x}_1$ and $\mathbf{x}_1$. Then it means that $\mathbf{l}$ is orthogonal to both $\mathbf{x}_1$ and $\mathbf{x}_2$, that gives:

$$\mathbf{l} = \mathbf{x}_1 \times \mathbf{x}_2 \tag{2.8}$$

This principle of that the properties of that point and lines in homogeneous coordinate are interchangeable is called the ”Duality principle” and is an important property of homogeneous coordinates.

2.3 Kinematics

In this section, it will be presented the kinematic tools that are needed to describe the different frames and the transformations between them. The theory in this section is taken from (Egeland and Gravdahl, 2002, p. 209-242).

The coordinate-free vector can be written as

$$\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^\top \tag{2.9}$$

The scalar product between two coordinate-free vector is

$$\mathbf{u}^\top \mathbf{v} = |\mathbf{u}||\mathbf{v}| \cos\theta \tag{2.10}$$

where $|\cdot|$ is the lenght of the vector and θ is the angle between the two vectors.

The vector cross-product is given by

$$\mathbf{u} \times \mathbf{v} = \hat{\mathbf{n}}|\mathbf{u}||\mathbf{v}| \sin\theta \tag{2.11}$$

where $\mathbf{n}$ is a unit vector that is orthogonal to both $\mathbf{u}$ and $\mathbf{v}$ and defined so that $(\mathbf{u}, \mathbf{v}, \hat{\mathbf{n}})$ forms the right-hand system.

A reference frame A can be described by three unit vectors that are orthogonal to each other, i.e.

$$\hat{\mathbf{a}}_1, \hat{\mathbf{a}}_2, \hat{\mathbf{a}}_3 \tag{2.12}$$

where $\hat{\mathbf{a}}_i^\top \hat{\mathbf{a}}_j = 0$.

A coordinate-free vector $\mathbf{v}$ described in reference frame A and frame B is

$$\mathbf{v}^a = \begin{bmatrix} v_1^a & v_2^a & v_3^a \end{bmatrix}^\top \quad \mathbf{v}^b = \begin{bmatrix} v_1^b & v_2^b & v_3^b \end{bmatrix}^\top \tag{2.13}$$

Then coordinate transformation from frame B to frame A is given by

$$\mathbf{v}^a = \mathbf{R}_b^a \mathbf{v}^b \tag{2.14}$$

where $\mathbf{R}_b^a$ has to be a element in $SO(3)$ to be called a rotation matrix.

Consider the rotation in Figure 2.2, this rotation is a simple rotation about the y-axis. In Figure 2.2 the y-axis is equal, $\hat{\mathbf{a}}_2^\top \hat{\mathbf{b}}_2 = 1$, to in both frames, which leads to that the other unit vector is orthogonal to each other:

$$\hat{\mathbf{a}}_2^\top \hat{\mathbf{b}}_1 = \hat{\mathbf{a}}_2^\top \hat{\mathbf{b}}_3 = \hat{\mathbf{a}}_1^\top \hat{\mathbf{b}}_2 = \hat{\mathbf{a}}_3^\top \hat{\mathbf{b}}_2 = 0 \tag{2.15}$$

And the x-axis and z-axis have the relationship

$$\hat{\mathbf{a}}_1^\top \hat{\mathbf{b}}_1 = \cos\theta, \quad \hat{\mathbf{a}}_3^\top \hat{\mathbf{b}}_3 = \cos\theta \tag{2.16}$$

$$\hat{\mathbf{a}}_3^\top \hat{\mathbf{b}}_1 = \sin\theta, \quad \hat{\mathbf{a}}_1^\top \hat{\mathbf{b}}_3 = -\sin\theta \tag{2.17}$$

The rotation matrix is then given by

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.18}$$

And the same goes for simple rotations about x-axis and z-axis.

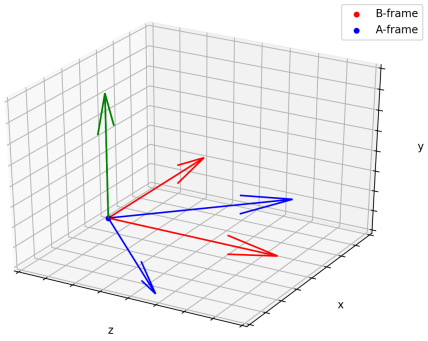


Figure 2.2: Simple rotation from frame A to frame B about y-axis

As for the simple rotations rotate about a fixed axis it can be shown that for all rotations it exists a unit vector the rotation is about

$$\hat{\mathbf{k}} = \hat{\mathbf{k}}^a = \mathbf{R}_b^a \hat{\mathbf{k}}^b \tag{2.19}$$

Given that the rotation from frame B to frame A about $\hat{\mathbf{k}}$ is given by θ, then the rotation matrix is

$$\mathbf{R}_b^a = \mathbf{R}_{k,\theta} = \mathbf{I} + \hat{\mathbf{k}}^\times \sin\theta + \hat{\mathbf{k}}^\times \hat{\mathbf{k}}^\times (1 - \cos\theta) \tag{2.20}$$

To describe the relative pose, position, and orientation, there need to be included a translation between the two frames. Given that the rotation between frame B and frame A is given by $\mathbf{R}_b^a$ and the relative position of frame A in frame B is $\mathbf{r}_{ab}^a$, then the transformation is given by

$$\mathbf{T}_b^a = \begin{bmatrix} \mathbf{R}_b^a & \mathbf{r}_{ab}^a \\ \mathbf{0}^\top & 1 \end{bmatrix} \in SE(3) \tag{2.21}$$

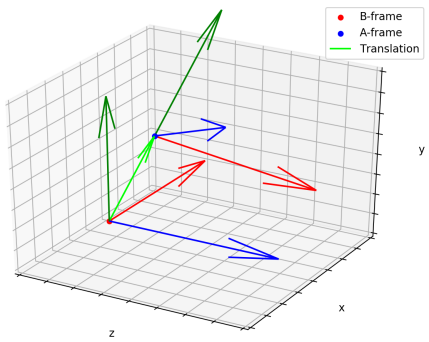


Figure 2.3: The relative pose of frame A in frame B

To apply a transformation matrix to a vector there is convenient to use a homogenogus representation of the vector, section 2.2. The transformation will then simply be:

$$\mathbf{v}^a = \mathbf{T}_b^a \mathbf{v}^b \tag{2.22}$$

2.4 Convex-hull

The Convex-hull of a set of points that forms a smallest convex polygon that the points either is on the boundary or is encapsulated by the polygons, (Cormen et al., 2009, p. 1029-1039). It can be thought of like a rubber band that surrounds a set of nails hammered into a plane. Convex-hull is widely used in many applications, such as box wrapping, obstacle avoidance, pattern recognition, and many more. Follows the many different applications there exist many solutions to find the Convex-hull, one of the is Quickhull, Barber et al. (1996). The Quickhull algorithm follows a divide and conquers approach similar to that of Quicksort.

The algorithm is outlined in Algorithm 1. Also, in Figure 2.4, there is an example of how Quickhull is used on a set of points. In **(b)** the to initial points, the left and right most, is found. In **(c)** - **(e)** the Findhull() function is performed to find the segments and vertices.

Also, in **(e)**, the termination of the algorithm is reached.

Algorithm 1: Quickhull

Input: Set of points, P
Result: Convex-hull, CH

$CH = \{\}$
Function `Quickhull(`P`)`**:**
    $A \leftarrow$ Find the left most point
    $B \leftarrow$ Find the right most point
    $CH \leftarrow A, B$
    Create a segment S_1 that is all the points that is the subset to the right of the line oriented $A - B$. Do the same for the segment S_2 that is oriented $B - A$.
    Findhull(S_1, A, B)
    Findhull(S_2, B, A)
    return CH
Function `Findhull(`S, P_1, P_2`)`**:**
    if $|S| == 0$ **then**
        return
    $C \leftarrow$ Find the point farthest away from line $P_1 - P_2$
    $CH \leftarrow C$
    Create a segment S_0 that inside the triangle P_1, P_2, C
    Create a segment S_1 that is all the points that is the subset to the right of the line oriented $P_1 - C$. Do the same for the segment S_2 that is oriented $C - P_2$.
    Findhull(S_1, P_1, C)
    Findhull(S_2, C, P_2)
return CH

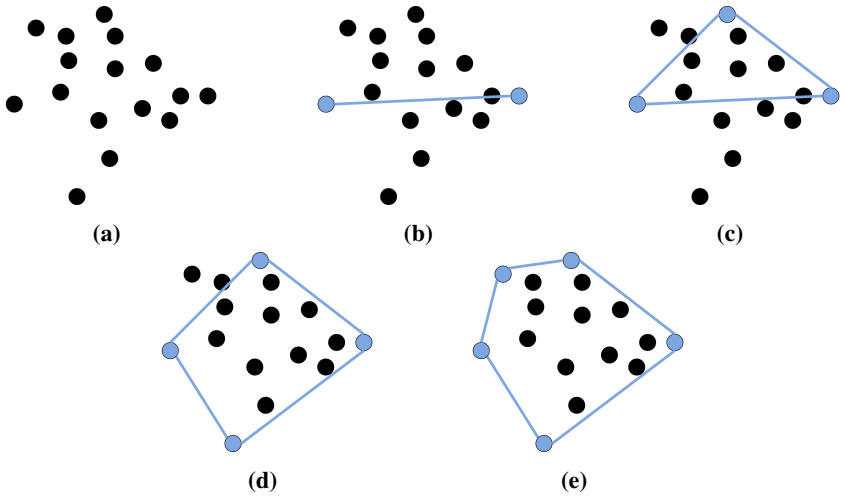
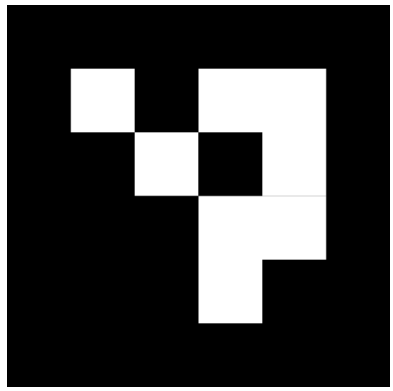


Figure 2.4: Example of Quickhull algorithm on a set of points

2.5 ArUco markers

Garrido-Jurado et al. (2014) was the one that induced the ArUco marker. ArUco marker is a binary square fiducial marker which is commonly used in many applications, such as augmented reality and robot localization, for pose estimation. With a single marker, it is possible to detect the pose by the correspondence with its four vertices. Garrido-Jurado et al. (2014) presents an algorithm for generating configurable marker dictionaries, an automatic method of detecting the markers and correction possible errors, and lastly a solution to occlusion problem in augmented reality. Now the method of detecting the Aruco marker and how to estimated the pose, will be reviewed.



The detection method in Garrido-Jurado et al. (2014) consists of an image analysis part, and marker identification and error correction part. Where the last part is their main contribution and makes it spacially robust. The method goes as follows:

- Image segmentation: Will thresholding be done on the gray-scale image. For computational reasons, they have chosen to use a local adaptive thresholding approach. This because they saw that a method like the Canny edge detector was to slow for real-time purpose.
- Contour extraction and filtering: All contours are extracted from the thresholding image. This is done by first approximate all the polygons from the contours. Since there is know that a marker consists of four vertices, it will discard all polygons that do not consist of four vertices.
- Marker Code extraction: Before it can identify the code in each marker, there needs to be done some preprocessing. First, the perspective is removed by estimating the homography matrix. Then binary thresholding is performed with Otsu's method. Lastly, the binary image is divided into a grid. Each of the cells in the grid is assigned either 0-1, determined on the major pixel. Now all markers that do not have a 0 around the border.
- Marker identification and error correction: At this point, it is necessary to determine which of the markers that are ArUco markers and which that is the environment. This is done by looking up each every rotation, four, in the set of all possible markers(dictionary). The ones that are in the set is determined as a valid marker.

2.6 Model fitting

This subsection is taken from the specialization project, Vasdal (2018)

2.6.1 Hough transform

The primary goal of the Hough transform is to check all feasible models that fit the data. To compare all the data points to all possible models is computational very expensive. Hough transform do this in the opposite way. Instead of check all points to every model, it finds all models that fit the points. Parametric models, models that can be described with a finite number of parameters, is ideal for this kind of transform.

Given a line, it can be described as:

$$y = a_0 x + b_0$$

All points that satisfy that equation is part of the same line with parameters a_0 and b_0. The Hough transform will map all points from image space through Hough space

$$b = y_0 - a_0 x$$

All points in image space will be transformed to a line in Hough space, and at the point where the parametric values have an intersection, they will have the same parametric model in that point, i.e., for lines the same line. It a simple example is shown in Figure 2.5.

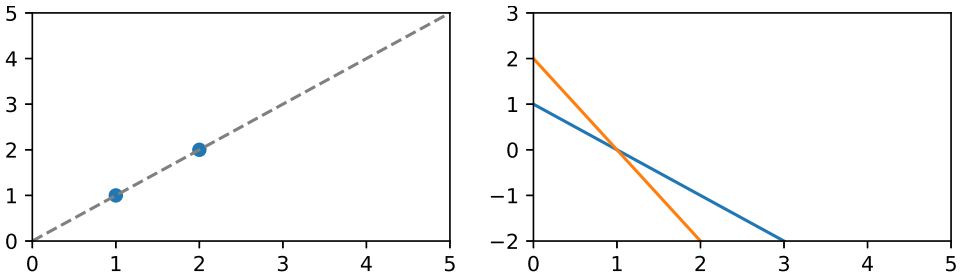


Figure 2.5: To the left there is two points, $(1, 1)$ and $(2, 2)$, in image space. To the right there is both the the Hough transform of both points into Hough space. They are represented as two lines, $b = 1 - a$ and $b = 2 - 2a$. The intersection between this two lines, point (1,0), is the line, gray dashed line, that goes through both points in image space.

This was a simple example. In an image, we will have more data and also noise. This will then lead to a more complex Hough transform, and the parameters for one line can deviate. I.e., we do not have one intersection between parametric values. To find the parametric parameters that fit the most points, we will have voting for each point inside a bin. The bins that have more votes than a given threshold is considered a line.

The method described above is made for parametric models, but it has also been made a generalized Hough transform. This method was introduced in Ballard (1981) and with this type of Hough transform, we can use it to find arbitrary object described with its model. E.g., if we want to find the position of an object in the image is transformed into finding the transformation parameters to place the object into the image.

2.6.2 Least-Squares fitting

Least-Squares fitting is a method for fitting models to a dataset. It is based on finding the parameters to the model that the absolute distance from the model to the date is minimal.

For a dataset of n points, where $\mathbf{x}_i$ independent variables and y_i dependent variables, for $i = 1, ...n$. The model that will be fitted is given by $f(\mathbf{x}_i, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the model parameters. The residual, $r_i = y_i - f(\mathbf{x}_i, \boldsymbol{\theta})$, tells that the error between the measurements and the model is. The Least-Squares solution is given when

$$\min_{\boldsymbol{\theta}} J = \min \boldsymbol{\theta} \sum_{i=0}^{n} r_i^2$$

For a linear model the dependent variables is given by $\mathbf{y}$, independent varabels is given by $\mathbf{x} = [\mathbf{x_1}\ \mathbf{x_2}\ ...\ \mathbf{x_k}\ 1]^\top$ and parameters $\boldsymbol{\theta} = [\theta_1\ \theta_2\ ...\ \theta_k\ b]^\top$, where b is a bias. This Least

squares problem can then be rewritten as

$$\min \boldsymbol{\theta}(\mathbf{y} - \mathbf{x}_i^\top \boldsymbol{\theta})(\mathbf{y} - \mathbf{x}_i^\top \boldsymbol{\theta})^\top$$

Solution to this is given when

$$\frac{\partial J}{\partial \boldsymbol{\theta}} = 0 \implies \frac{\partial J}{\partial \boldsymbol{\theta}} = -2\mathbf{x}\mathbf{y} + 2\mathbf{x}\mathbf{x}^\top \boldsymbol{\theta} = 0$$

The minimal value Least-squares solution to a linear model is given when

$$\boldsymbol{\theta} = (\mathbf{x}^\top \mathbf{x})^{-1}\mathbf{x}^\top \mathbf{y} \tag{2.23}$$

where

$$\mathbf{x}^\dagger = (\mathbf{x}^\top \mathbf{x})^{-1}\mathbf{x}^\top \tag{2.24}$$

is defined as the psudo-invers.

An important note is that the least-squares solution does try to find the parameters that fit all data points, also when they could lay outside the model.

2.6.3 Random Sample Consensus

RANSAC, Random Sample Consensus, is a method that is widely used for model fitting and was first introduced in Fischler and Bolles (1981). This method is based on the knowledge that there exist some points within the dataset that describe the model, and there are outliers from this model. The assumption that RANSAC takes is that the outliers of the model is random and votes against the model. The inliers to the model all agree that the models are right and vote for the model. Therefore the goal is to find the right points so that the model can be computed.

The method is done by an iterative process. First, it will be drawn a sample uniformly random from the dataset. This sample consists of a minimum number of points that are needed to fit the model. For a line, it needs two, and for a plane, it needs three. Then the residual of the model needs to be measured. The points that are inside a given threshold from the model is considered as inliers. This process continues until a high probability that the model has the right parameters.

In Figure 2.6, there is a comparison between a RANSAC fitted line and Least-squares fitted line to data points with outliers. Here it is clear that the Least-squares solution is affected by the outliers and estimates the wrong line.

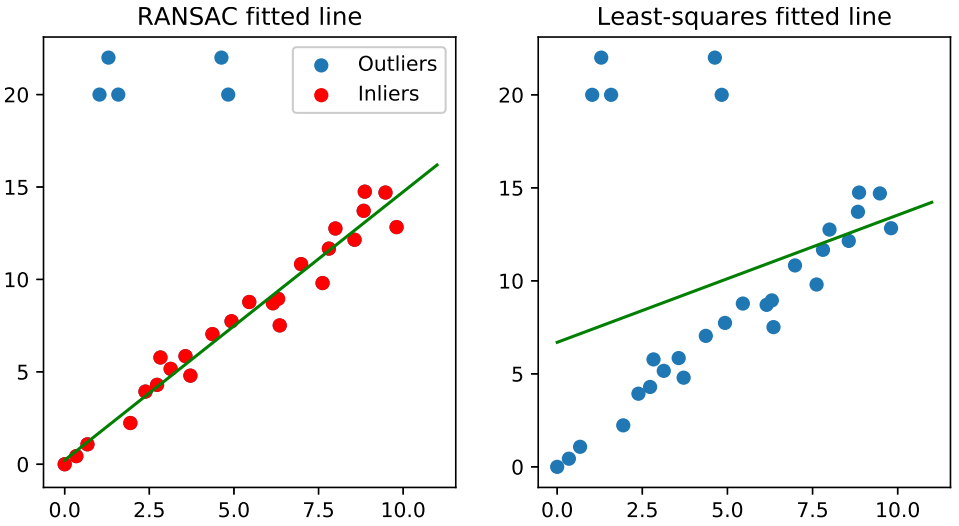


Figure 2.6: Comparison between RANSAC line fitting and Least-squares line fitting.

2.7 Bin Packing Problem

In the chapter, it will try to give a better understanding of the whole Bin packing problem, BPP. This by in the first section provide a general mathematical description of the BPP. In the second section, it will be given an overview of the different kind of 3D-BPP that is commonly addressed. In the third section, it will be reviewed some of the most important work that has been done in the field of 3D-BPP. Lastly, a concrete formalization of the Robotic bin packing problem will be addressed.

2.7.1 Definition of the Bin Packing Problem

In order to state and solve the problem for this thesis, there needs to be gain more knowledge about the BPP. Therefore there will define the general 3D-BPP. The definition presented here is inspired by Hu et al. (2017). The main difference is that in Hu et al. (2017) want to minimize the total surface area of one bin. While in the general 3D-BPP there will be a minimization of the number of bins needed for packing a set of boxes.

$$\min \qquad \sum_{m=0}^{M} y_m \tag{2.25}$$

Subject to

$$a_{im} \cdot (x_i - x_j) + L \cdot s_{ij} \leq L - a_{im} \cdot \hat{l}_i \tag{2.26a}$$

$$a_{im} \cdot (y_i - y_j) + W \cdot s_{ij} \leq W - a_{im} \cdot \hat{w}_i \tag{2.26b}$$

$$a_{im} \cdot (z_i - z_j) + H \cdot s_{ij} \leq H - a_{im} \cdot \hat{h}_i \tag{2.26c}$$

$$0 \leq x_i \leq L - \hat{l}_i \tag{2.26d}$$

$$0 \leq y_i \leq W - \hat{w}_i \tag{2.26e}$$

$$0 \leq z_i \leq H - \hat{h}_i \tag{2.26f}$$

$$\hat{l}_i = \delta_{i1} l_i + \delta_{i2} l_i + \delta_{i3} w_i + \delta_{i4} w_i + \delta_{i5} h_i + \delta_{i6} h_i \tag{2.26g}$$

$$\hat{w}_i = \delta_{i1} w_i + \delta_{i2} h_i + \delta_{i3} l_i + \delta_{i4} h_i + \delta_{i5} l_i + \delta_{i6} w_i \tag{2.26h}$$

$$\hat{h}_i = \delta_{i1} h_i + \delta_{i2} w_i + \delta_{i3} h_i + \delta_{i4} l_i + \delta_{i5} w_i + \delta_{i6} l_i \tag{2.26i}$$

$$a_{im} a_{jm} \cdot (s_{ij} + u_{ij} + b_{ij}) + (1 - a_{im} a_{jm}) = 1 \tag{2.26j}$$

$$\delta_{i1} + \delta_{i2} + \delta_{i3} + \delta_{i4} + \delta_{i5} + \delta_{i6} = 1 \tag{2.26k}$$

$$s_{ij}, u_{ij}, b_{ij} \in \{0, 1\} \tag{2.26l}$$

$$\delta_{i1}, \delta_{i2}, \delta_{i3}, \delta_{i4}, \delta_{i5}, \delta_{i6} \in \{0, 1\} \tag{2.26m}$$

$$y_m \in \{0, 1\} \tag{2.26n}$$

$$a_{im} \in \{0, 1\} \tag{2.26o}$$

Variable	Type	Meaning
y_m	Binary	bin m or not
a_{im}	Binary	item i is place in bin m or not
L	Continuous	the length of the bins
W	Continuous	the width of the bins
H	Continuous	the height of the bins
x_i	Continuous	LLB coordinate of the item i in x-axis
y_i	Continuous	LLB coordinate of the item i in y-axis
z_i	Continuous	LLB coordinate of the item i in z-axis
s_{ij}	Binary	item i is in the left side of item j or not
u_{ij}	Binary	item i is under item j or not
b_{ij}	Binary	item i is in the back of item j or not
δ_1	Binary	orientation of item i is front-up or not
δ_2	Binary	orientation of item i is front-down or not
δ_3	Binary	orientation of item i is side-up or not
δ_4	Binary	orientation of item i is side-down or not
δ_5	Binary	orientation of item i is bottom-up or not
δ_6	Binary	orientation of item i is bottom-down or not

Table 2.1: Decision variables for the general 3D-BPP

$$\sum_{m=0}^{M} a_{im} = 1 \tag{2.26p}$$

where the description of the decision variables is given in table 2.1. As mention, this definition describes the minimization of the number of bins needed to pack a set of boxes. The constraints that are given in the definition help finding a feasible solution. Constraints (2.26a), (2.26b), (2.26c), (2.26j) will guarantee to not get any overlap between item i and j. Constraints (2.26d), (2.26e), (2.26f) will guarantee that the placement of the item i is inside the bin m. Lastly (2.26k) will help with rotation the item i and (2.26g), (2.26h), (2.26i) will give the length, width and height of the item i after it is rotated.

To reduce the 3D-BPP to a 2D-BPP, it can be done by removing one dimension from the problem. E.g. in the z-axis, then the decision variables needs to be changed: $H = \infty$, $h_i = \infty$, $z_i = 0$, $b_{ij} = 0$, $\delta_2 = 0$, $\delta_2 = 0$, $\delta_4 = 0$, $\delta_5 = 0$ and $\delta_2 = 0$. This will result in that (2.26c), (2.26i) and (2.26f) will be removed.

In the case of reduction to 1D-BPP, it only needs to take the capacity into account. Therefore all decision variables that describe the position, orientation, and relationship between items will be removed. In the end, the three constraints, (2.26n), (2.26o), (2.26p) will be kept, and merge the other constraints:

$$\sum_{m=1}^{m} \sum_{i=1}^{I} a_{im} l_i < L \tag{2.27}$$

this constraint does that all constraints are independent of unnecessary decision variables.

2.7.2 Different type of 3D Bin Packing Problems

In industrial applications, there is not only minimizing the number of bins that are of consideration. By doing some modifications to the problems, it can use many of the similar methods to find a solution to other packing problems. Wäscher et al. (2007) gives an overview of many the different types of packing problems. Common for all of these problems is that there are given two sets of elements:

- a set of small items.
- a set of large objects.

Divide some or all items into one or more subsets. Assign each subset into one of the larger objects such that the geometric conditions hold, i.e., the small items lies within the larger object and do not overlap each other.

If the placement of all the items gives an optimum to a given objective function, a solution is found. The solution to the problem may result in using one or all large objects, and some or all small items.

To concretize the different problems Wäscher et al. (2007), gives some further criteria that could define this thesis packing problem. These criteria will concern the assortment of small items and large objects and the objective function.

The assortment of small objects is divided into three types of size distribution. First is identical small items, where all the items have the same size and shape. Second is weakly heterogeneous items, where the items are grouped into relatively few classes, for which the items are identical concerning shape and size. The third is strongly heterogeneous items, the set of small items are by the fact that only a few elements have the same shape and size. If they are equal, they are treated as individual elements. For all types of distribution shapes identical shaped and sized object with different orientations are treated as different kinds of items.

The assortment of large objects is dividing the two main groups. The first group, the set of a large object, consists of only one large object. This can be a fixed-sized object, or its extension may variable in one or more dimensions. The second group, the set of large object is more the one, and the large objects are of fixed size. However, the sizes within the set do not have to be equal. It can be used the same analogy of describing the size distribution as for the assortment of small objects.

Considering the objective function, it will eighter formulate the problem as a maximization or minimization problem. The difference between the two cases lies in the accommodation in the large objects. In the case of maximization, there is not given that the set of large objects is sufficient to accommodate all the small items. For the minimization, the large object is sufficient to accommodate all small items, but all small items are to be assigned to a selection of the large objects.

This gives a general understanding of how the different primary type of packing problem that could be met. In many applications, it is necessary to give even more constraints to the problem. Bortfeldt and Wäscher (2013) gives an overview of the state-of-the-art for container load problems where other types of constraints taken into account. This is e.g., weight, orientation, stacking order, positioning, stability, and complexity.

2.7.3 Solving 3D Bin Packing Problems

To find a solution to the BPP is not in any way trivial. As mentioned the problem 1D BPP is considered NP-Hard, not solvable in polynomial time (Garey and Johnson, 2002, p. 124-127), Also, 3D BPP is, therefore, strongly NP-Hard. In this section, there will be conducted a brief studied of three types of approaches to solving BPP, exact method, heuristic methods, and lastly deep reinforcement methods.

The study of exact methods for BPP is mostly done in late 1990 to 2007. Martello et al. (2000) gives an exact algorithm to solves the problem of a single BPP. They have developed a branch-and-bound algorithm that guarantees the optimal solution. As well they give a good. In Martello et al. (2007) they continue the development and makes the algorithm to work on a general BPP. Fekete and Schepers (1997, 2004) are using graph-theoretic characterization for mapping the relative position of the items, where it is feasible to pack. By using this and with good heuristics for dismissing infeasible items, they have implemented an exact algorithm that can solve 2D- and 3D-BPP.

The exact methods can guarantee optimal solutions; however, are usually very timing consuming even on moderate datasets. The use-case is, therefore constrained. A heuristic algorithm, cannot guarantee optimality, gives a good solution to the BPP with much less computational time. Scheithauer (1991) was the first to propose an approximation algorithm for the 3D-BPP. They reduce the problem to multiple 2D-problem so that they stack each layer with a branch-and-bound algorithm. Faroe et al. (2003) propose a heuristic algorithm based on guided local search heuristic. By starting with an upper bound obtained from a greedy heuristic, the algorithm reduces the number of items until the upper bound is equal to the lower, or it has exceeded the time. There have been multiple suggestions for using Tabu search algorithms. Lodi et al. (1999) developed a general Tabu Search framework for 2D-BPP. By defining a search scheme and a neighborhood which is generalized, they were able to write the 2D-BPP framework to a 3D-BPP with few changes in Lodi et al. (2002). Crainic et al. (2009) also have used Tabu Search to both reduce the number of bins needed to pack the items and optimize the packing of each bin.

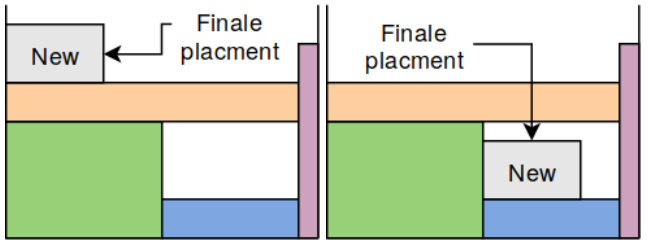
For the case of this thesis, it will be a robotic manipulator that will pack the items. A natural to choice then is to pack the items one-by-one. In comparison to Scheithauer (1991) and Lodi et al. (2002) which is packing layer-by-layer, place all items into a layer then pack. The drawback for this is that there if the input forces fragmentation of the loading surface there will degrade the rest of the solution. A solution for using the one-by-one packing there is common to divide the decision into:

1. the sequence in which the items are packed into the bins.

2. the strategy of selecting the placement of the item.
3. item orientations.

The methods that have mostly been investigated in this thesis are using Genetic algorithms(GA) for optimizing the stacking sequence, and a heuristics approach for selecting the placement and orientation of the items. Karabulut and İnceoğlu (2004) was one of the first to introduce GA in BPP, and at time of publication, it was reported to perform well. In Kang et al. (2012), Li et al. (2014), and Gonçalves and Resende (2013), they use the encoded packing sequence in the chromosomes and GA to evolve the sequence towards the optimal solution. Kang et al. (2012) use this on a single bin problem and reports that they find a near-optimal solution within an acceptable amount of time. Li et al. (2014) encodes both the packing sequence and the container load sequence and use that to find a solution for general BPP. Gonçalves and Resende (2013) uses GA both to find the stacking sequence and to find the orientation for each box on a general 2D- and 3D-BPP. They report their method is equally or outperformance then other approaches.

To find the placement of the items, all methods use heuristics packing. The work of Baker et al. (1980) starts one of the first most notable works for packing strategy. They introduced the bottom left, BL, packing strategy for 2D BPP. This places the item in the bottom left of the bin, Figure 2.7a. After that many improvements were made in 2D packing strategies, among them was Hopper (2000) who introduced bottom left with fill, BLF, Figure 2.7b. Compared to BL, it also fill spaces that BL left out. In Karabulut and İnceoğlu (2004) proposed DBLF, deep bottom left with fill, this was an extension of BLF to the 3D-BPP. DBLF moved the item as deep into the bin (smallest z value), far as possible to the bottom (smallest y value), and finally as far to the left as possible (smallest x value), but at the same time fill the space.



(a) Bottom left **(b)** Bottom left with fill

Figure 2.7: 2D packing huristics.

In the most recent packing strategies, it is common to use the concept of Empty Naximum Space, Lai and Chan (1997), to manage the packable area. This by defining the all feasible space-objects as a different cuboid. Doing this, it can reduce expensive computational operations such as reduce the number of space-objects by removing all space object underneath the dimensions of the items. Methods such as Kang et al. (2012) and Li et al. (2014) use EMS together with DBLF to perform the placement of each of the items. While in

Gonçalves and Resende (2013) use a distance to the front top right corner to determine which space-object in EMS that should be chosen for packing.

The most reaches in the field of BPP is done with deep learning. The first publication that tried to solve 3D-BPP was Hu et al. (2017). A neural network called Pointer-Net, Vinyals et al. (2015), inspired them. This network had previously been used with good results to solve the Traveling salesman problem, Bello et al. (2016). Hu et al. (2017) used deep reinforcement learning on the Pointer-Net to solve the packing sequence, while it uses a heuristic packing strategy with EMS. Later Hu et al. (2018) came up with a multi-task framework that was able to generate packing sequences and the orientation of the items. For generating the sequence, they used deep reinforcement learning, while for the orientation, it was supervised learning. The most important contribution they did was that they did two parts together that was previously have been treated separately. Lastly, Laterre et al. (2018) have been inspired by work like AlphaZero to learn from Self-Play Reinforcement Learning. By letting neural network learn by playing itself, it outperforms, e.g., heuristic algorithms.

2.7.4 Robotic Bin Packing Problem

To better understand the problem that will be addressed in the thesis, it will be given an example before the formal definition.

In modern production, line boxes could be an important part of the system. In many of these applications, the content that will be produced is known, but the sequence which the boxes are arriving in is not given. In such a system, there is very hard to palletize the boxes with a robotic manipulator. To do this there, the system can be made such that the robotic manipulator can choose n to pick for packing. The decision to choose which of the boxes to pick and where to place it will be called the Robotic bin packing problem. Formerly it will be stated as:

Definition 1 (Robotic Bin Packing Problem)**:**
Given a set of items and one container, with a size of $(W \times L \times H)$*. Find the subset,* I*, of boxes that minimizes the waste space:*

$$W \cdot L \cdot H - \sum_{i=0}^{I} l_i \cdot w_i \cdot h_i$$

Subject to

- *All the items in the subset lies entirely within the container*
- *All the items do not overlap each other*

When all items, N_{items}*, that should be packed is known, but only* N_{pick} *items available for each pick and.*

Chapter 3

Method and implementation

3.1 Introduction

In this chapter, two parts of a pipeline, Figure 3.1, for solving a fully automated box palletizing system have been developed. In section 3.2 there will be developed a method for detection of multiple boxes within a point cloud. This method includes three parts; preprocessing part to transform the point cloud into a define reference frame, a part that does the detection of the boxes and lastly a features extraction part. The last section of the chapter, section 3.3, deals with the Robotic bin packing problem, RBPP. It will be developed a new type of packing heuristic for robotic packing applications, a Generic algorithm that solves a single container Bin packing problem, BPP. And, a method for RBPP which choose which box to pack next and where to pack it.

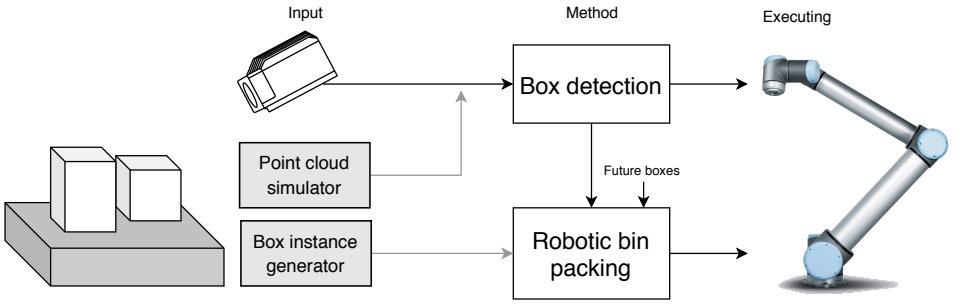


Figure 3.1: Overall pipline over the fully automated box palletizing system

Two tools have been developed in order to test the methods: a simulator for box detection, and an instance generator for robotic packing. This allows the possibility of testing the

methods independently of each other, and to control the difficulty of the test.

3.2 Box detection in a point cloud

The are three steps that are needed to detect all of the boxes in a point cloud. First is point cloud preprocessing, it will is to give as good base before any detection is done. Second is detection, where points that are part of a box is separated from the point cloud. Lastly, the pose and size are estimated.

Figure 3.2: Parts of the box detection

3.2.1 Point Cloud preprocessing

The point cloud preprocessing consist of two parts, as illustrated in Figure 3.3:

First parts will detect all planar surfaces that are considered as noisy in the point cloud. It will then calculate a transformation for each of the surfaces. Which does that the removal of the planar surfaces can be done by filtering along an axis.

The second part will calculate the relative pose, $\mathbf{T}_B^C$, between the Camera-frame and a defined frame called Base-frame. This Base-frame will be a unique description for all part of the system. All the object present in the system will have its relative pose be described in Base-frame. To use this reference-system it needs to be computed the relative pose between the Camera-frame and the Base-frame.

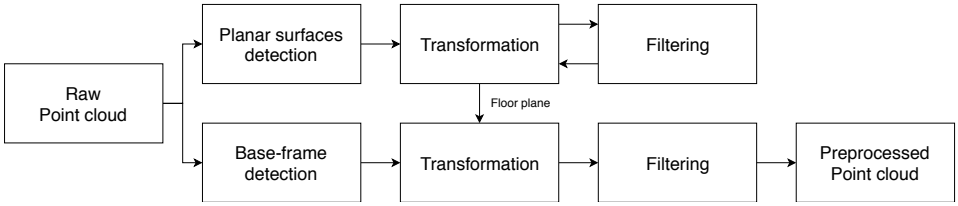


Figure 3.3: All steps done in the preprocesing of the point cloud

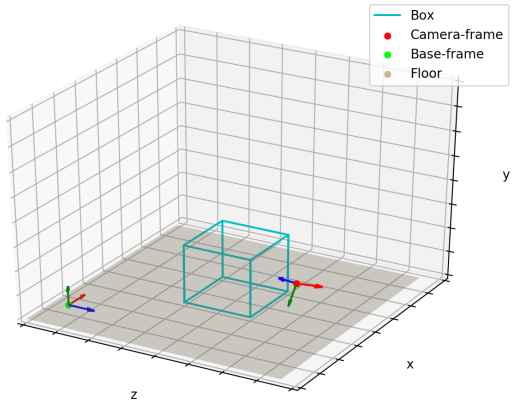


Figure 3.4: The two frames in our problem

All the transformations calculated in this section can be done offline by assuming a stationary camera and scene. These detections are expensive, and by doing them offline the total expense will be reduced with a significant amount.

In the rest of this section, it will be reviewed in more details all the in the preprocessing, as illustrated in Figure 3.3.

Plane detection with RANSAC

To estimate the planar surfaces in the point cloud, it needs to be a method that is robust and possible to detect multiple surfaces. Hough transform, section 2.6.1, have successfully been used for detection of 3D-shapes even in data with a high proportion of outliers. However, as Illingworth and Kittler (1988) reports due to low efficiency or high memory consumption, it is not that suited. A pure plane fitting with least-square, section 2.6.2, would not work since there is a possibility of multiple planes and outliers. The method that is commonly the most method for model fitting with a high proportion of outliers is RANSAC, section 2.6.3. In the case of this problem Schnabel et al. (2007) reports that it can be used for detection of primitives shapes, e.g., planes, in a point cloud.

The general method of RANSAC is described in 2.6.3. To use RANSAC, there need to be defined two properties about the model. These properties are: how to compute the model and the residual to the model. To describe points and plane, it will be used homogenous coordinates, as described in section 2.2. This because of its relationship between points and plane in three-dimensions.

To compute the plane-model, we only need the right null space of equation (2.4) with a sample size of 3 linear independent points. The residual will be computed as the difference of the distance from the plane to origo and the distance of the point to Origo, i.e.:

$$\epsilon = \left| \frac{\pi_4}{\|\boldsymbol{\pi}_{1,2,3}\|} - \|\mathbf{x}\| \right| \tag{3.1}$$

Where $\boldsymbol{\pi}$ is the estimated plane, and $\mathbf{x}$ is the point that is computed the residual to the plane.

In order to know when to stop searching for new shape in the RANSAC algorithm, there needs to be define a termination condition. From Schnabel et al. (2007) given a termination condition that will calculate the number of iterations so that with a probability there will be detect a plane within the iterations, it goes as follows. Given a point cloud, $\mathcal{P}$, with N points. Within the point cloud, there is a plane that consists of n points. The probability of detecting the plane in a single pass is:

$$P(n) = \binom{n}{3} / \binom{N}{3} \approx \left(\frac{n}{N}\right)^3 \tag{3.2}$$

The probability of a succsessful detection a succsessful candidate after s itterations is given by:

$$P(n, s) = 1 - (1 - P(n))^s \tag{3.3}$$

Solving for s tells us the number of itterations T required to detect a plane of with n points with a probability $P(n, T) \geq p_t$:

$$T \geq \frac{\ln(1 - p_t)}{\ln(1 - P(n))} \tag{3.4}$$

The detection done in this thesis will be done offline, therefore, probability p_t could be set very high to be sure that the planes that have been detected will be fitted very well. This since there is no need to think about computation time.

As described in Schnabel et al. (2007) a refitting step can be done after the detection is done. This step is consist of doing a least-square, section 2.6.2, fitting of all the point accounted as an inlier. This optimizes the geometric error for the planar surface.

Base-frame detection

In order to get a description of the Base-frame that will be detectable by a camera, there were chosen to use an ArUco marker, section 2.5. The purpose of using ArUco marker is that it is robust when detecting the marker and very easy to use. One of the most important properties for the ArUco marker is that the relative pose between the camera and the ArUco marker can be estimated by minimizing the reprojection error of the corners of the marker. However, since the quality of the point cloud captured by the Zivid-camera is of such high quality, there is of interest to utilizing the information within the points.

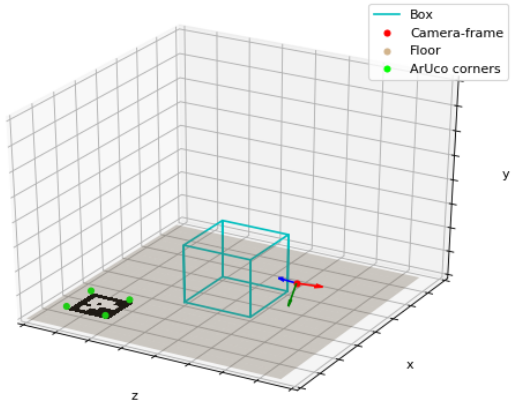


Figure 3.5: The ArUco marker placed in the sceen to define the Base-frame.

The detection of the ArUco marker is done in the RGB-image, and therefore the corners of the ArUco marker are given in the pixel-frame. Since the point cloud outputted from the Zivid-camera is organized, it is trivial to find the corresponding point in the point cloud. One challenge with using ArUco marker is that the Zivid-camera will not detect any points on dark surfaces. Therefore it is necessary to use an ArUco marker with inverted intensity and do an invention of the colors before detection.

There are two features that will be calculated from the ArUco marker. It will define the center of the Base-frame, and it will calculate the rotation such that the point cloud is aligned with the Base-frame.

The orientation of the Base-frame is computed from the edges of the ArUco marker, see Figure 3.6. This is done in three steps. First, it will be found the normalized vector, $\hat{\mathbf{v}}_i$, of each edge. Secondly define a unit vector, $\hat{\mathbf{w}}_i$, that is in the principal axis and the direction of $\hat{\mathbf{v}}_i$. Thirdly the orientation of each of edges is given by the dot product between $\hat{\mathbf{v}}_i$ and $\hat{\mathbf{w}}_i$. Out of this, there will be four different angles. The orientation of the marker is computed as the median of the four orientation.

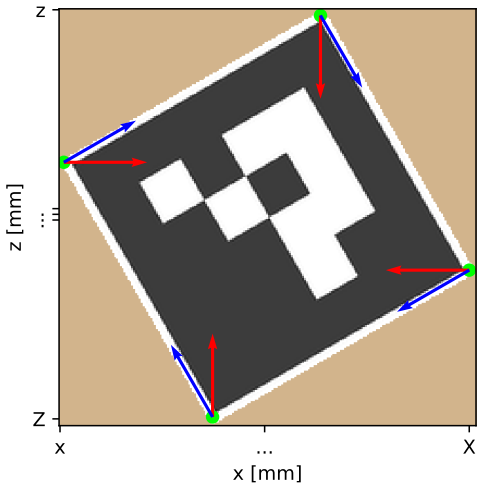


Figure 3.6: Illustation of how the orientation of a ArUco marker is estimated. The green dots are corners, blue vectors, $\hat{\mathbf{v}}_i$, the vector between corners and the red vectors, $\hat{\mathbf{w}}_i$, are the principal axis of $\hat{\mathbf{v}}_i$.

Transformation

As stated at the beginning of this section, there where two different types of transformations that will be calculated. The first transform is used to align a planar surface with a known axis, such that the point in the planar surface could be filtered out

$$\mathbf{T}_c^p = \begin{bmatrix} \mathbf{R}_c^p & \mathbf{r}_{pc}^p \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{3.5}$$

where $\mathbf{r}_{pc}^p$ is a arbitrary point on the surface. $\mathbf{R}_c^p$ is the rotation ti align the surface normal, $\hat{\boldsymbol{\pi}}_{1,2,3}$, from the planar surface with the y-axis in Base-frame, $\hat{\mathbf{b}}_2$. To compute this rotation it will be used the axis-angle representation (2.20)

$$\mathbf{R}_c^p = \mathbf{R}_{k,\theta} = \mathbf{I} + \mathbf{k}^\times \sin\theta + \mathbf{k}^\times \mathbf{k}^\times (1 - \cos\theta) \tag{3.6}$$

Where $\mathbf{k} = \hat{\boldsymbol{\pi}}_{1,2,3} \times \hat{\mathbf{b}}_2$ and $\cos\theta = \hat{\boldsymbol{\pi}}_{1,2,3}^\top \hat{\mathbf{b}}_2$.

The second transform is to align the whole point cloud with the Base-frame

$$\mathbf{T}_c^b = \begin{bmatrix} \mathbf{R}_a^b \mathbf{R}_c^a & \mathbf{r}_{bc}^b \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{3.7}$$

where $\mathbf{R}_c^a$ computed equally the plane algiment that where did in (3.6). $\mathbf{r}_{bc}^b$ is the centroid of the Base-frame calucalted in from the ArUco marker. And the last rotation is computed as the simple rotation (2.18) about the y-axis with the orientation computed from the ArUco marker.

3.2.2 Detection

In this section, there will be developed a method for the detection of multiple boxes in the point cloud. The parts of the point cloud that is considered to be a box will be further processed before the feature will be extracted in the next part.

Clustering

In this stage, all points that are part of a box will be extracted from the point cloud. Rodriguez-Garavito et al. (2018) suggest using Mean-Shift clustering. The two arguments why they choose Means-Shift was for its capabilities to detect a variable number of boxes and for having presented good results as an unsupervised classification. In order to use Means-Shift, there needs to be searched in the calculated direction in which the points should be shifted. To do search operations in a point cloud should be kept to a minimum since these operations are expensive. Methods such as Kd-Tree can reduce the search time. The method that has been used in this thesis is the Connected Component. Connected Component is that it utilizes the point cloud from the Zivid-camera is organized point cloud, and do not need to do any search operations. Which makes it much faster than Mean-shift. The general performance is pretty good, but with the wrong camera setting, it can have some weaknesses.

We will now review how Connected Components will cluster the point cloud. The origin of the method is from Trevor et al. (2013). As mentioned the method requires that the point cloud is organized. This does that it can be access a points neighborhood only by its pixel-coordinates. Which makes it possible to find the neighborhood in constant time.

The method works by partitioning the point cloud into segments. This by making labeling, $L(u, v)$, of each point in the point cloud. The points that corresponding to the same segment will have the same label, i.e. $L(u_1, v_1) = L(u_2, v_2)$. To determine this, there have to be used a comparison function which takes two points as input and returns true or false based on how the function is defined.

$$C(P(x_i, y_j), P(x_n, y_m)) = \begin{cases} true\ \ if\ \ similar \\ false\ \ otherwise \end{cases} \tag{3.8}$$

If $C(P(x_1, y_1), P(x_2, y_2)) = true$ then $L(u_1, v_1) = L(u_2, v_2)$, else $L(u_1, v_1) \neq L(u_2, v_2)$.

In the case of this thesis, there have been used the Euclidean distance between each point as the comparison function, other possibilities could for instance be surface normals and pixel intensity. These methods have not been used, because: By using surface normals, the clustering will cluster each of the planner faces of the box as individual clusters. Also, it is expensive to compute surface normals. By using the pixel intensity will not work since there for example could be labels with a different color than the box. While with the only Euclidean distance, it does not consider these changes is surface.

The labeling algorithm starts by assigning the first valid point with the label 0. Compare the points in the first row and column with the specified comparison function and assigned

a label after that. The remainder of the points is treated by examining their neighboring points, left and above. If both neighbors have different labels, but the comparison function returns true for neighbors, then the labels to the two points should be merges. This is done by using the union-find algorithm. A second pass is done to merge labels such that it will reduce the number labels. This will finally return the labeled point cloud. The labels that contain fewer points, then a threshold will be excluded. Typically there is a significant gap between the labels that contain a box or not, so the threshold is easy to set.

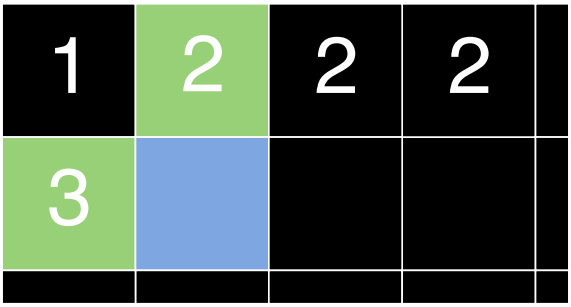


Figure 3.7: Example of two labeling points that should be merged. Current pixel is in blue without any label and green is the neighboring pixels. If the blue matches green, the labels should be merged. This will be done in the second pass using the union-find algorithm

Height calculation

All the features estimations will be done in the top-plane of the box. However, the method for separating the top-plane from the rest of the box depends on a good estimation of the height. Hence, the height has to be computed before the other features.

Since the point cloud is aligned with the Base-frame, any of the points on the top-plane will represent the height. The centroid of the box will be used as the initial point for calculation of the height. Because there are no points on the planes not facing the camera, the centroid will be located closer to the edges of the top-plane. The final height is represented as the mean of y-value of all the points within 5 cm.

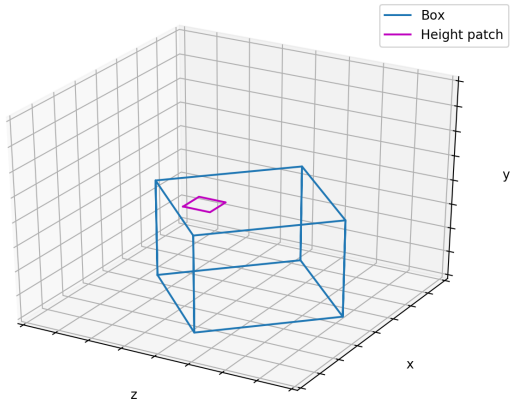


Figure 3.8: An example of a patch used for estimation of the height.

Top-plane seperation

Since the point cloud still consists of many points, there is expensive to do RANSAC and downsampling. Downsampling could may some important information that can been lost. The method that is suggested use the fact that the top-plane is co-planar planar to the ground plane.

To extract the top-plane, there is needed a threshold for what should be accounted for as inliers. There is common to the same constant threshold on both sides of the plane when extracting inliers. However, in this problem, there have been experienced that there are uneven surfaces on the top-planes. Having a constant threshold on both sides, there will include many points on the side walls and the whole top-plane, or there will have few points from the side walls and lose some information on the top-plane. When there are many points from the side walls, it will cause problems in the next step when the points are projected onto the floor plane. It will in the next step cause problems when the points are projected onto the floor plane. If there is an uneven surfaces on the side walls, there will be points from the side walls that will make padding around the projection. With an adaptive threshold this problem will not occur.

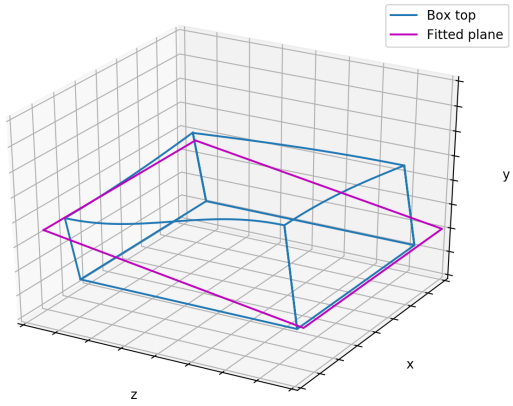


Figure 3.9: Illustation of the challage with seperation of the top-plane with an uneven surface.

In this appraoch, the prior knowledge of that there is more points in a horizontal slice of the top-plane of the box then in the middle of the box. The method starts by taking a horizontal slice with a predefined height 1 mm, see Figure 3.10. The number of points at this slice will be set as the reference of what that will be considered as a part of the side walls. The method will it iterate along the y-axis from starting from 10 mm below the height of the box. For each iteration, a new slice with a height of 1 mm will be extracted and compare it with the reference slice. If the ratio of points in the new slice is above a defined threshold, th_{slice}, it will be considered as the minimum value of the top-plane. All points in the cluster that are above the minimum value are considered a part of the top-plane.

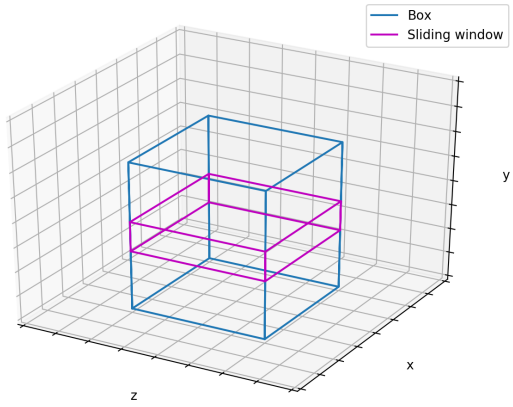


Figure 3.10: Sample slice of the boxes, used as an common value of points on the side plane.

Edges estimation

The features will be estimated from the edges of the top-plane. Since the top-plane is co-planar with the floor-plane, it will be projected onto the floor-plane. To compute the edges, it will be used Convex-hull, section 2.4, which is the smallest set of vertices that encloses all the points in the top-plane. The method used is Quickhull, Algorithm 1 11, which is implemented in PCL.

The Convex-hull have too many vertices that describe the top-plane. In this application, it is only needed to have the four edges on each side of the box. To reduce the number of vertices, Algorithm 2 is applied. This algorithm makes a vector between each vertex and then by comparing how much change in direction there is between the vectors there can be determined two properties. First, current vector direction change so much that it is part of a new edge, line 7 in Algorithm 2. Secondly, the current vector has the same direction as previous, and they are part of the same edge, line 9 in Algorithm 2. The vector with a length beneath a threshold, th_{lenght}, will not be considered as a possible part of an edge.

Algorithm 2: Edges around a plane

Input: Convex hull, CH
Result: Edges along plane

```
1  edges = {}
2  foreach vertex ∈ CH do
3      vector = vertex − vertex_next
4      if |vector| > th_lenght then
5          if vertex_idx > 0 then
6              θ = angle(vector, vector_prev)
7              if θ > th_add then
8                  edges ← vertex, vertex_next
9              if θ < th_merge then
10                 merges edges from vector and vector_prev
11         else
12             edges ← vertex, vertex_next
13             vector_prev = vector
14 return edges
```

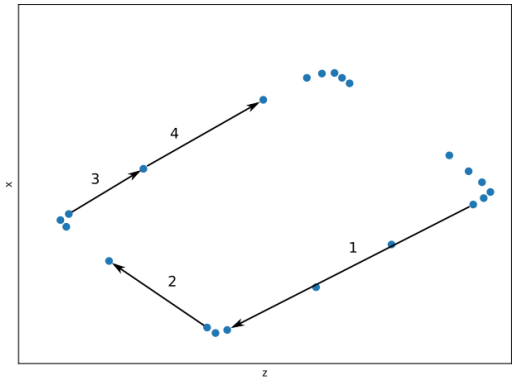


Figure 3.11: An examples of computation of edges from the Convex Hull (blue dots). The the first vector is counted as a full edge. When the second vector is computed it is accouted as a new edge. The two last vector will be merge and be a part of a edge

3.2.3 Feature estimation

Now the features that described that box can be estimated. All the features are described with respect to Base-frame. Therefore the orientation of the box is only α about the y-axis. The position, (x, h, z), is defined in the center of the top-plane. The size of the box is given as width, height, and length, (w, h, l). For both position in y-direction and the height, h, it is already calculated in section 3.2.2. The features are illustrated in Figure 3.12.

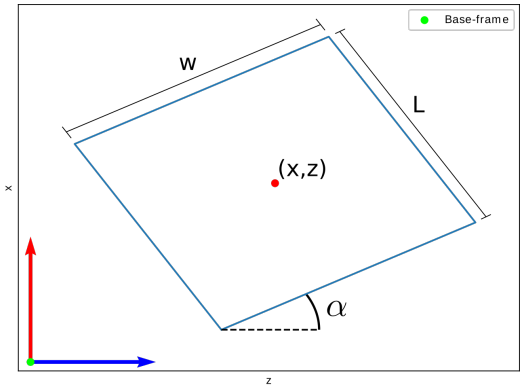


Figure 3.12: Features that will be estimate. Orientation, α, position, (x, h, z), size (w, h, l)

Position

To estimate the position, a common approach is to calculate the centroid to find the position. One of the disadvantages for calculating the centroid is that an assumption that the points are uniformly distributed over the plane has to be taken, which is not always the case. If a point cloud is captured from an angled oreintation, there will be a different density of points at different depths. Also, in the case of partwise occluded top-plane, there will be a part of the plane that is missing points.

To calculate the position in a more robust way, the edges have been utilized by paring each of the parallel edges and compute the mean line for both pair lines. Then the center of the box is calculated as the intersection of the mean lines.

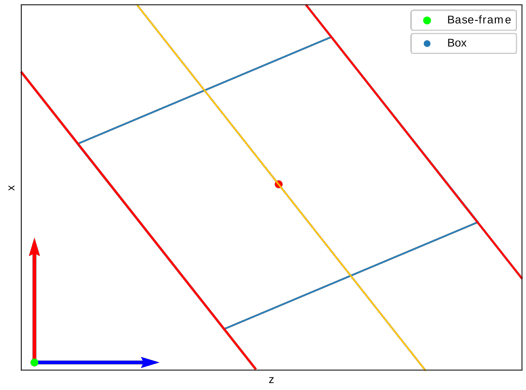


Figure 3.13: The lines of two parrallel edges is illustrated in red and the mean of these two lines is in yellow

Orientation

A common technique to calculate the orientation of a set of points is to use PCL. However, as for the centroid, it will compute the wrong orientation due to unevenly distributed points.

The orientation, α, is computed equivalent as there was done in the computation of the orientation of the ArUco marker in section 3.2.1. Instead of using the vector between each corner in the ArUco marker, there will use the vector on each edge in the edges. From these calculations, there will be four orientations. The final orientation is given by it the median of these four orientations.

Size

The width and length are given as the distance between two pairwise edges. It is computed with the line for each edge and then computes the distance between each of the lines.

$$\mathbf{x}^\top \mathbf{l}_1 = 0 \tag{3.9}$$

$$\mathbf{x}^\top \mathbf{l}_2 = 0 \tag{3.10}$$

$$d = \left| \frac{l_{1_3}}{\|\mathbf{l}_{1_{1,2}}\|} - \frac{l_{2_3}}{\|\mathbf{l}_{2_{1,2}}\|} \right| \tag{3.11}$$

3.3 Optimal Robotic bin packing

It this thesis there have been developed a method that will solve Robotic bin packing problem, section 2.7.4. Since the research on RBPP is not that well studied, the work of this method is based on the research done to solve BPP, section 2.7.3. BPP problems have been studied for many years and will give fundamental knowledge towards a solution to the problem.

In the RBPP, there are three classes of decision that need to be taken:

1. Packing, where should the given box be placed into the pallet.
2. Sequence, which will calculate the optimal sequence of boxes if they are packed one-by-one.
3. Picking, which of the boxes should be pack first.

All these decisions are strongly dependent on each other. If the Picking had been removed, the problem would be a BPP.

In the following sections, it will be reviewed how each of the decision will be solved.

3.3.1 Packing of boxes

The main two steps that are done in this section is that a new concept of Empty Supported Maximal Space, ESMS, that keeps track of the feasible space where boxes can be packed will be introduced. Moreover, the second step there will be used some heuristics to choose where in this feasible space, the box will be packed. In Figure 3.14 it is an overview of how the packing of one box on the pallet is performed. If a set of boxes should be packed, the procedure in Figure 3.14 is repeated for all the boxes. The boxes that do not fit into any space will be placed in an unpacked box list.

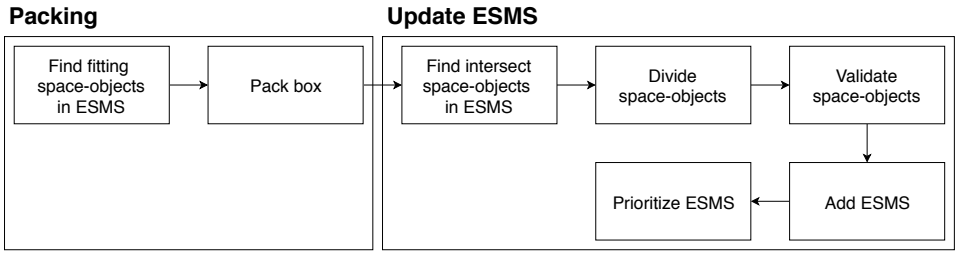
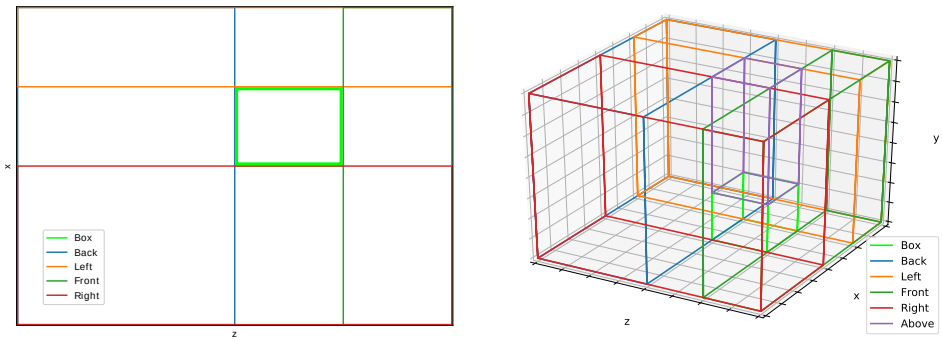


Figure 3.14: Overview of operations that is preformed when a box is packed



(a) 2D view seen down on XZ-plane

(b) 3D view

Figure 3.15: Maximal number of space-objects.

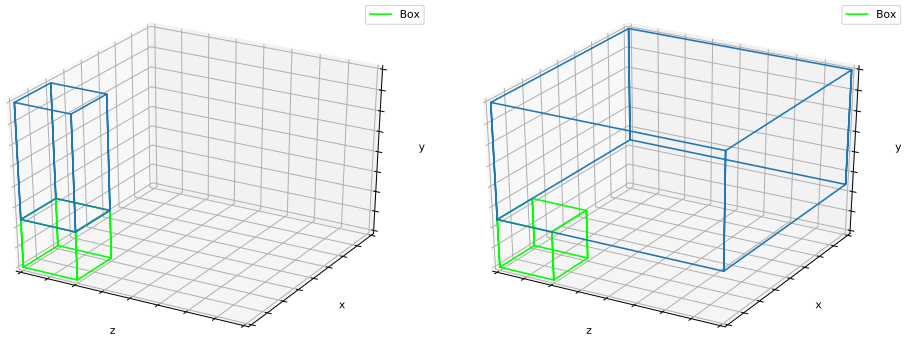
Empty Supported Maximal Space

ESMS, will managing which feasible space which will possible for packing boxes. This is done by dividing the unpacked volume into space-object, s_i, which says the maximum dimensions that could be packed into space. The idea is inspired by Lai and Chan (1997) who introduced the concept of Empty Maximal Space, EMS. Both of the methods are managing a feasible space where boxes could be placed. The difference between them is that in ESMS, guarantees that there always will be supported surface, ground, or another box, underneath the space.

In the initialization, a list will be declared, S, which will contain all the feasible spaces. The first space-object that will be inserted is created with coordinate in left-bottom-back, $(0, 0, 0)$, and have the the size is equal to the maximum packing volume, $(w_{max}, l_{max}, h_{max})$. When inserting a box onto the pallet, a update of S is needed. The update process starts by finding all space-objects that intersect with the box that should be packed. The intersecting space-objects are so updated by removing the displaced volume, i.e., dividing the space-objects into new feasible spaces. The maximum number of divisions of a space object is 5 in ESMS, i.e., back, left, front, right and above. A 2D example of how the space-objects will look like after an update step can be seen in Figure 3.15a. An in the 3D case, Figure 3.15b, there is a space above as well,

The search for intersecting spaces is a time-consuming operation. This search will be done both for placement of a new box, and to search for if a recently created space-object is inscribed by other space-object and thereby need to be added to S. There will be made some rules for which space-objects that will be added to S. This will do that the total number of space-objects will be reduced, and expensive operation like a search for intersection will be cheaper

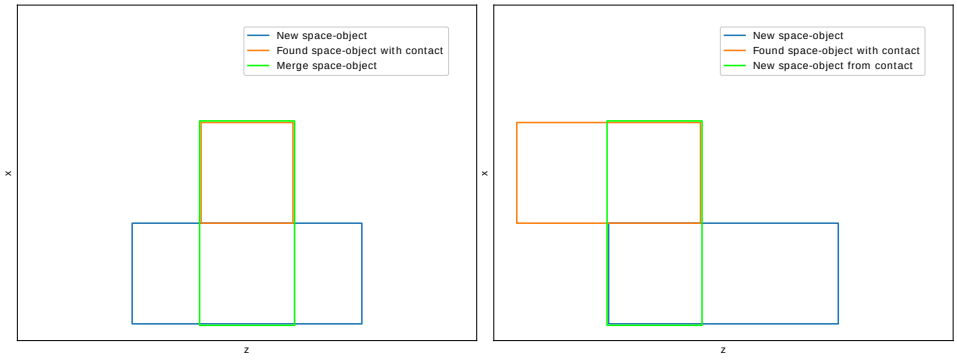
- If the newly created space-object volume is smaller than each of the remaining box volume, the space-object will not be add into S.
- If the dimensions, (w, l, h), of the space-object is smaller then each of the remaining



(a) ESMS, space above the box will be feasible packing area.

(b) EMS, the dissipated volum above is feasible packing area.

Figure 3.16: Difference between ESMS and EMS.



(a) Merge two space-objects.

(b) Create a new space-object in the touching points.

Figure 3.17: Merger of space-object that is created above a box

box dimensions, the space-object will not be add into S.

In Gonçalves and Resende (2013) they report that the reduction of computational is 60% for applying these rules, for most problem instances.

The difference between ESMS and EMS is when the space above the box is created. For EMS it will leave all the dissipated space as a space-object, see Figure 3.16b. These space-objects have no need for any merge of space-object, while ESMS will there is a need for this. Since the space-object above is only the supported be the box, it needs to be merged with neighborhood space-object as same height. The merger starts with a search through S for spaces that touches the newly created space-object. Depending on how the boxes touch each other, there will eighter merge the two space or create a new space-object in the touching points.

Packing heusterisic

The packing heuristic, section 2.7.3, that have been considered in this thesis is the packs the boxes one-by-one, a natural choice since there is a robotic manipulator that performance the packing. Ideally, the heuristic choose will pack the boxes as compacted solution as possible. The heuristics are used to sort S every time space-object is created, see Figure 3.14.

The heuristics used in this thesis is Deep Bottom Left with Fill, DBLF, that Karabulut and İnceoğlu (2004) proposed. It has been used in multiple other publications such as Kang et al. (2012) and Li et al. (2014) with good results. DBLF moved the item as deep into the bin (smallest z value), far as possible to the bottom (smallest y value), and finally as far to the left as possible (smallest x value), but at the same time fill the empty space.

When searching for a box position, only the position of the feasible space-objects, $S_{feasible}$, in S. $S_{feasible}$ is all space-objects where one or more orientations of the box fit inside (number of orientations depends on the problem statement).

3.3.2 Stacking sequence

In most of the BPP, it is possible to change the sequence of the boxes that should be packed. In this case, all boxes to be stacked are known, but the robot is only allowed to chose between a random subset of them. This makes it harder to optimize, and based on the sequence of available subsets, it may be impossible to pack the bin optimal. Most of the research is on BPP, this knowledge had to be transformed to the RBPP domain. In the next section, it will be shown that RBPP consists of solving multiple BPP for each box pick.

Genetic algorithms have gained much acceptance for solving optimization problems. It is a biology-inspired optimization method that will evolve during the optimization. GA cannot guarantee an optimal solution, but in general, it returns a satisfactory result in a reasonable time. In this section, the search for the optimal stacking sequence with the use of GA will be reviewed. This GA is based on the work of Li et al. (2014), although compared to the RBPP, they have multiple containers while RBPP have a single pallet.

Starting with defining some terminology:

- Population, set of trail stacking solution: $P_k = \{s_k^p; \quad p = 1, \ldots, P\}$.
- Chromosomes, represents a candiadate stacking solution: $s_k^p = \{1, \ldots N\}$.
- Parent, member of current population: s_k^p.
- Child, member of next population: s_{k+1}^p.
- Generation, successively created population: P_{k+1}.
- Fitness, measurment of how much spaces a stacking solution utilizes: $F_k^p = C\left(s_k^p\right)$.

With the terminilogy defined there will be a reviewed of the whole optimization. Overview of the framework can be found in can be found in Figure 3.18.

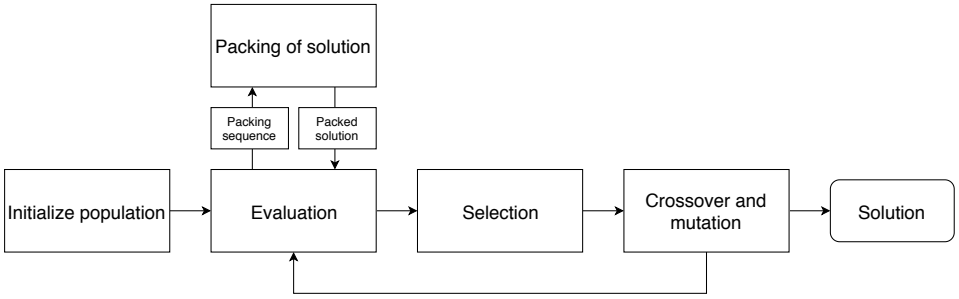


Figure 3.18: The framework of the Genetic algorithm.

Initialize population

Li et al. (2014) and Wang and Yanjie Chen (2010) give a suggestion to how the GA could be initialized, and is what will be used. They suggested to generate some special chromosomes in the first generation. They have done observations that the bigger boxes should be packed into the pallet early. The initial population, P_1, will have the four first chromosomes sorted by the volume, length, width, and height. The rest of the population in the first generation will be generated randomly.

Selection

GA is based on natural evolution, where the chromosomes which performs well are used for learning in the next generation. First, a sort of all of the chromosomes in descending order by their utilization of space, F_k^p, is done. The first E will be chosen by elitism which directly proceeds to the next generation, P_{k+1}. Now missing $|P_k| - E$ chromosomes for the next generation are missing. The last chromosomes that will be selected have to be selected first for joining a mating pool. A tournament selection makes the selection of chromosomes. Two randomly chromosomes from the population is selected in each round of the tournament selection. With a probability, $prob_t$ the better one is added into the mating pool; otherwise, the weak one is added. The chromosomes in the mating pool are paired up as parents. For each of the parents, there is a probability of $prob_c$ that they goe directly into the next generation; otherwise two offsprings are generated through a crossover.

Crossover and mutation

To explore more of the solution space there is proposed to do crossover and mutation. Crossover does this by inherent properties of well performance chromosomes and mutation is doing this by adding unexplored space into the solution.

Two parents, s_k^1 and s_k^2, are needed to perform crossover. These will generate two offsprings, s_{k+1}^1 and s_{k+1}^2. To find the the genes that are inherit to the first child, s_{k+1}^1, two cutting points i and j will be randomly pick. The genes in s_k^1 between $i+1$ and j will be copied to s_{k+1}^1. The missing genes is gather by sweeping through s_k^2 and circularly the genes that are missing into s_{k+1}^1, starting from $j+1$. To find the second child,s_{k+1}^2, there is simply just to change the two parents. In Figure 3.19 there is an example of crossover.

The mutation is performed on newly generated offsprings with a probability of $prob_m$. If the offsprings are selected, there will be randomly selected two genes that will swap their positions.

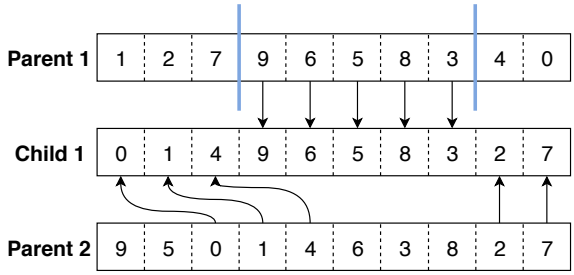


Figure 3.19: Crossover preformed between two parents.

3.3.3 Robotic bin packing

The big difference from BPP to RBPP, section 2.7.4, is the constraints on the possibility to change the sequence that the boxes arrive. In RBPP all, N_{items}, the boxes that should be packed is know, but it is only possible to choose between a few, N_{pick}. For these kinds of problem, it is not given it is possible to find a solution to the problem, due to the hard constraint regarding the order.

An example of the problem can be seen in Figure 3.20. In this example, the robot manipulator needs to choose which of the three boxes that need to pick up and pack on the pallet. This decision depends on the current solution on the pallet. However, equally important, it also is dependent on all the future boxes that will arrive later.

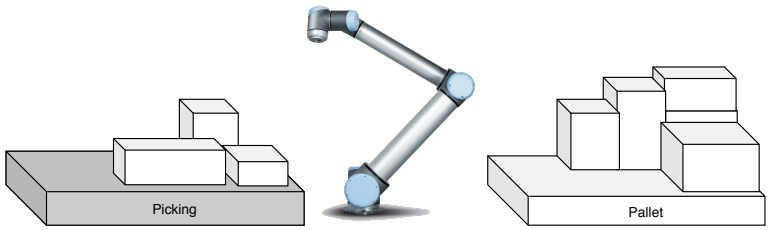


Figure 3.20: Illustration of how the decition for the robot can look like, with $n_{pick} = 3$

The method that will be suggested to solve this RBPP problem is illustrated in Figure 3.21. For each of the N_{pick} boxes available to be picked, temporary pack the box and remove it from the list of not-packed-boxes. Then, find the best stacking sequence for each of the remaining not-packed-boxes and store the N_{cost} best costs after the cost is calculated for all the available boxes. Based on these costs, pick the box that will have the best cost in the future. This box will then be packed on the pallet with the heuristics packing strategy in section 3.3.1, that includes updating the ESMS.

Then the problem is reduced to find a method to calculate the possible future cost. In the last section, 3.3.2, the GA search to find the optimal stacking sequence for a given set of boxes. If this method is given the list not-packed-boxes, it will calculate the optimal stacking sequence for the future boxes that will arrive. Since the GA is search by trying multiple attempts, there is possible to output the N_{cost} best costs. This can then be used in the method above to decide on which box to pick.

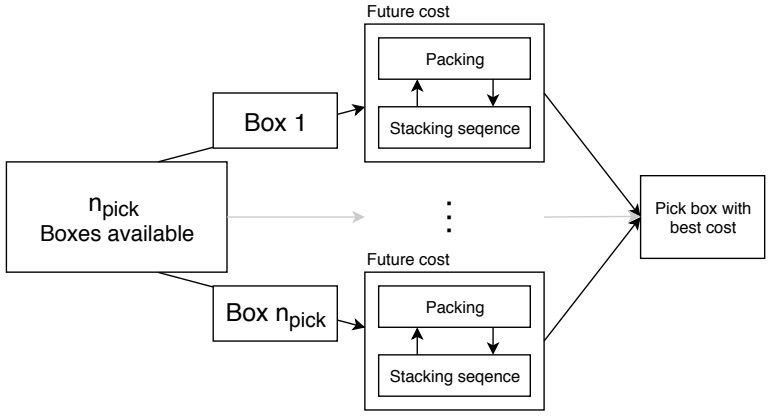


Figure 3.21: Pipeline of the Robotic bin packing. The methode calucalted the future cost of available boxes and picks the one with best future.

Chapter 4

Result and Analysis

4.1 Box detection

In this part, we will present the conducted experiments and the following results for the box detection method described in 3 chapter. The goal of the experiments is that we want to map the performance and uncertainties of the method.

4.1.1 Data

To get sufficient data for testing the performance of the method, there is not enough with data from a 3D-camera. It will only output the point cloud and to get a ground truth, we have to measure the features manually. This is a time-consuming task, inaccurate, and some of the features are impossible to measure. For this reason, we have developed a point cloud simulator, Appendix A. With the simulator we can:

- Add all types of objects into the scene.
- Place the object freely around in the scene, regarding both rotation and translation.
- Move the camera, regarding both rotation and translation.
- Add noise to the scene.
- Output a point cloud as seen from a 3D-camera, while knowing the exact pose and size of each object in the scene.

Also, we can test the performance on much more data and have control over each of the feature at every experiment.

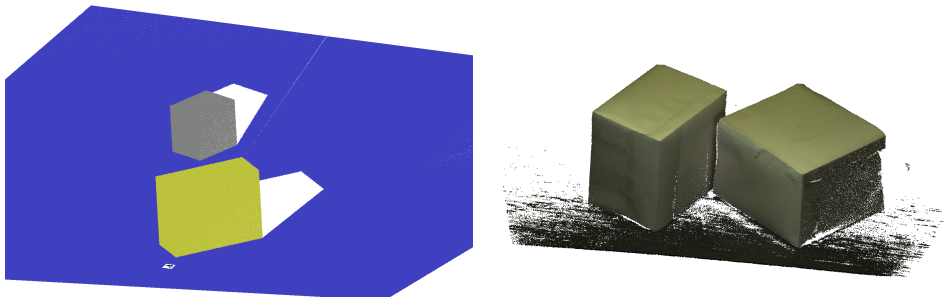


Figure 4.1: To the left is the point cloud from the simulator, and to the right is the point cloud from the Zivid camera. The point cloud from the Zivid camera is without ArUco markes, and have only been used for size estimation.

4.1.2 Cloud preprocessing

In our experiment, we will only focus on instances where we have one planar surface that supports the boxes. The center and orientation of the Base-frame is given as an ArUco marker and will also be present in all of the experiment. To get an accurate measurement of the performance, we need a ground truth. Therefore all the results are obtained on simulated data and some visual samples from the Zivid-camera.

The experiment conducted has the following properties:

- Random camera orientation between $30°$ and $70°$ about x-axis and $-40°$ and $40°$ about the y-axis.
- Number of point clouds: 50
- Number of boxes in each point cloud: 0
- Total number of boxes: 0

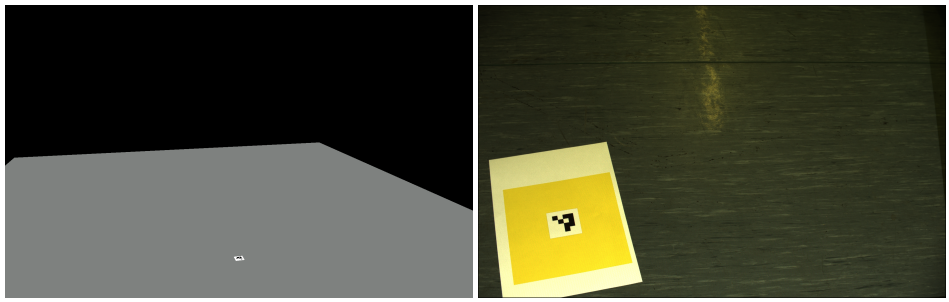


Figure 4.2: Input to ArUco detection from simulator and Zivid-camera.

The results from the experiment can be seen in Table 4.1. As we can see, the computated orientation has a perfect match. Important to notice is that for 7 of the captured point

clouds, we did not find the ArUco marker. So with these point clouds, we cannot estimate the orientation about the y-axis.

From the Zivid-camera, we captured 8 point clouds with different positions for the ArUco marker, and we did detect all. In Figure 4.3, we have an example of one of the output point clouds.

	y-axis	k-axis
Mean error	0.00°	0.058°
Standard deviation error	0.00°	0.00°

Table 4.1: Results camera orientation estimation.

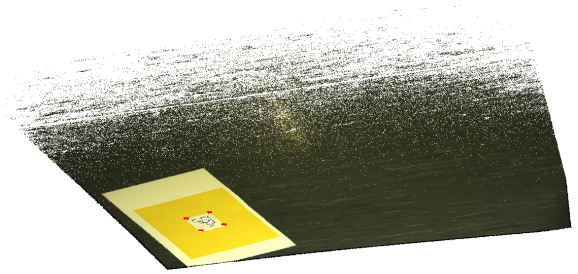


Figure 4.3: An example of the output of the ArUco, the red dots are the corners.

4.1.3 Preprocesing of clusters

Clustering

The clustering part of the method makes it possible to do detection of multiple boxes. This makes it a critical part of the method. To evaluate the performance, we could make the use of that the simulator gives us which points in the point cloud that represents the boxes. This is a good metric, but the data quality from the simulator is to ideal so we will return the correct clusters of the point cloud each time. The experiment in the clustering part will be conducted on data from the Zivid-camera, and the result will be the number of detected boxes.

The experiment conducted have the following properties:

- Random placed boxes in the scene.
- Number of point clouds: 22
- Number of boxes in each point cloud: 2
- Total number of boxes: 44

In 20 of these point clouds, we detected all the boxes. In this set of point clouds do we have samples where we have a large and short distance between the boxes, occlusion for another box and boxes partly is outside the point cloud. Two examples of this can be seen in Figure 4.4 and Figure 4.5.

We have also observed two cases where the clustering fails. Both of the cases have to do with that we are using Connected Components with Euclidean distance as a comparator. In the first case, Figure 4.6, we have physical contact between the boxes. The neighboring boxes will have small Euclidean distance in the contact area, and then be determined as the same cluster. For the second case, Figure 4.7, we have a discontinuity in the surface. The neighboring points will then have infinite Euclidean distance and then not able to bind the two sides if the discontinuity.

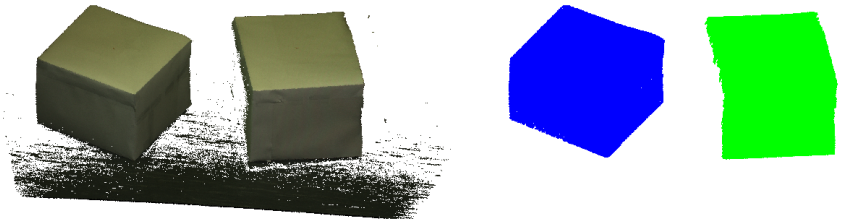


Figure 4.4: Successful clustering of two boxes

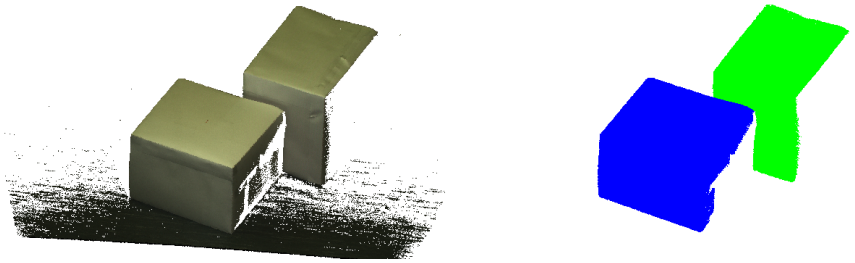


Figure 4.5: Successful clustering of two boxes, with occlusion

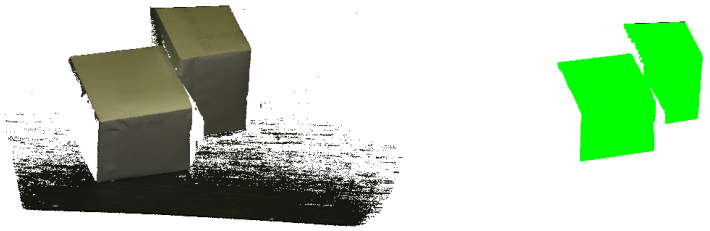


Figure 4.6: Failed to cluster the two boxes due to physical contact between the boxes

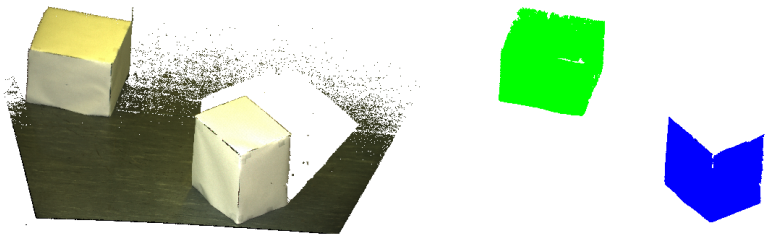
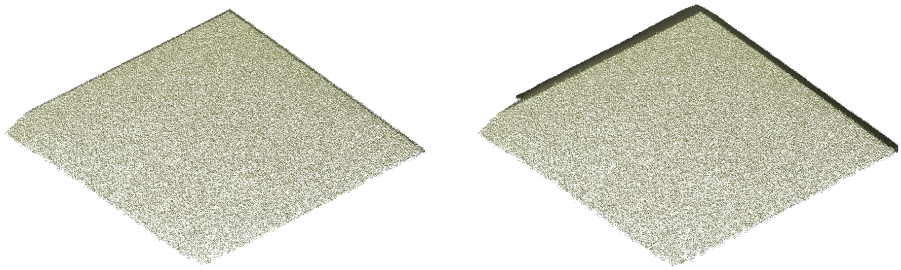


Figure 4.7: Failed to cluster the two boxes due discontinuity in the surfaces

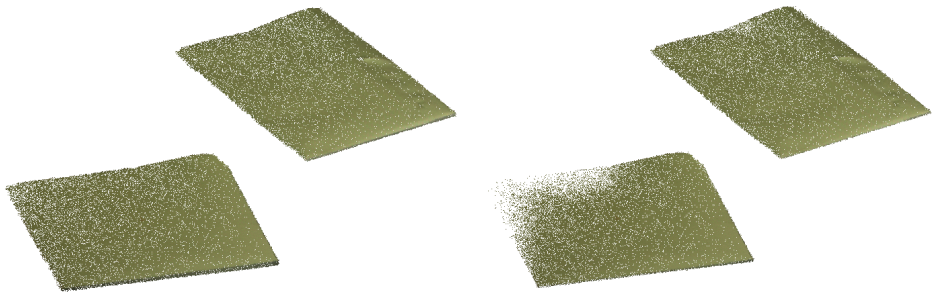
Top-plane extraction

With a constant threshold, two errors can occur. The first error is that we have included too many points from the side-planes, to large distance threshold. When projecting this into the top-plane equation, there will be padding around the projection, Figure 4.8. The second error is due to that the distance threshold is too small. For some top-planes it could be sufficient distance, but since we do not have any prior information about the upcoming planes this distance can be too small. In Figure 4.9 we have one plane that the distance threshold fits well, and one that has removed parts of the plane.



(a) No error **(b)** Error

Figure 4.8: Error due to projection of points from side-planes.



(a) No error **(b)** Error

Figure 4.9: Error due to that we remove to many points from the top-plane.

With an adaptable distance, we will avoid these two errors. The decision of the distance threshold is decided on that a change in the ratio of points within a sliding cross section along with the box. With a sample of the point clouds captured with the Zivid-camera, we have made a plot visualizing the change of points within the sliding cross-section, Figure 4.10. We can see a significant change for all of the samples and that there are different distances from the height for the box. In our case, we have set the termination condition to be a ratio of $th_{slice} = 2$.

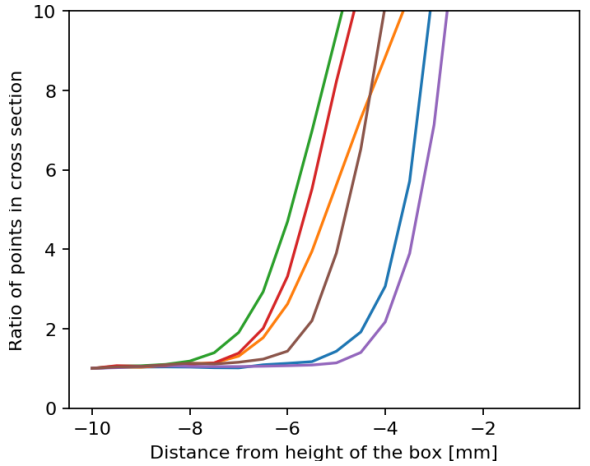
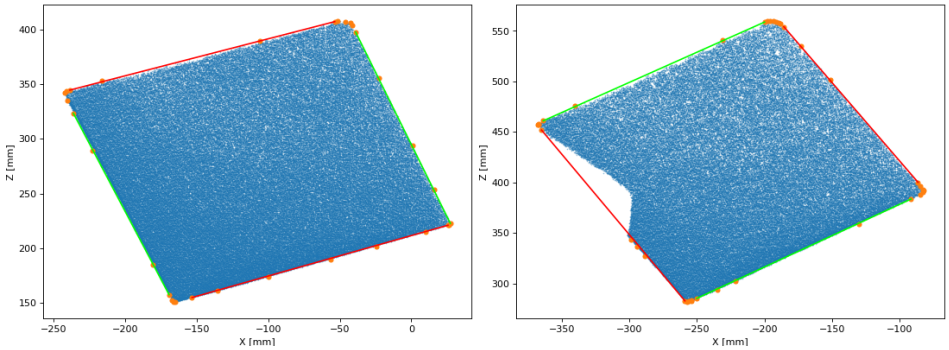


Figure 4.10: Ratio of inliers of the sliding cross section.

Edges

Now we will look into how the computation of the edges from the Convex hull will look like. This is dependent on three thresholds that is defined in beforehand, the thresholds we had was $th_{length} = 15mm$, $th_{add} = 10°$ and $th_{merge} = 80°$. In Figure 4.11 have we picked out two instances of interest from data obtained by the Zivid-camera. Both

Figure 4.11a and Figure 4.11b have very good results. The edges follow perfectly outside of the planes. The most interesting result is that we can compute the edge when we have occlusion. Which does such that there is possible to compute the position and orientation without the use of the centroid and PCA.



(a) Perfect edge computation. **(b)** Edge computation on a occluded point cloud.

Figure 4.11: Examples of the edge computation. The orange dots represent the output from the convex hull, red and green lines the computed edges.

4.1.4 Height calculation

The height calculations are being used both as a feature in height and position along y-axis. Also, it will be used as a parameter in the next steps in our method, which makes it important for multiple reasons. Since boxes height is possible to measure, it can be done experiment with real data from Zivid-camera.

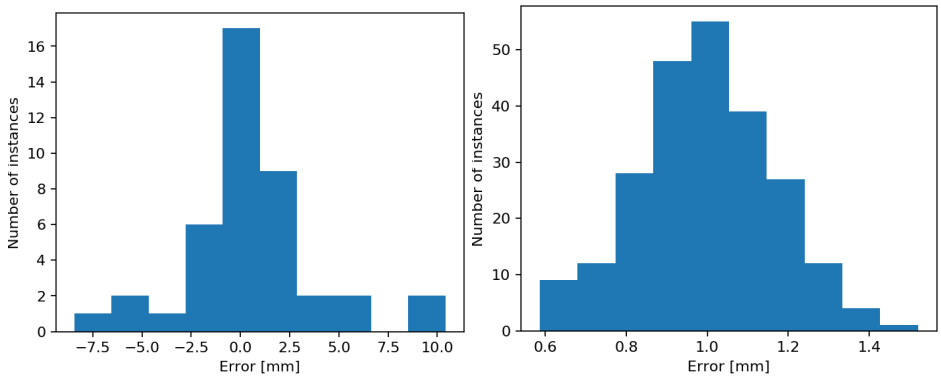
The experiment conducted with the Zivid-camera has the following properties:

- Two fixed sized boxes
- Number of point clouds: 20
- Number of boxes in each point cloud: 2
- Total number of boxes: 40

As we can see in Table 4.2 and 4.12 the result from the Zivid-camera have fairly good results. The largest part of the estimation is within 2.5 mm from the correct height. Also, for the Zivid-camera, we have a different type of measurement error then can occur with the simulator. E.g., on the surfaces of the box faces the simulator will have a perfect plane with only Gaussian noise. While for the case of Zivid-camera experiments, we are working with real-world objects where we have uneven surfaces where only the mean of all the points in a face could be a plane. The centroid of the top-plane, where we calculate the height, can have a small deviation from what we have measured. Also, the height of the boxes is measured manually, and will there will also be a measurement error. We have also done an experiment with 235 boxes from the simulator.

	Zivid	Simulator
Mean error $[mm]$	0.66	1.00
Root mean square error $[mm]$	2.73	1.01
Standard deviation error $[mm]$	3.42	0.17

Table 4.2: Results height calculation.



(a) Zivid-camera **(b)** Simulated

Figure 4.12: Distribution of errors in height calculation.

A question that could be asked is why we do not use centroid to the position in x and z as well. This can be seen in Figure 4.13 where we have two cases where the centroid of the points corresponding to the true center top-plane and one case where it deviates. To use the centroid as position, there has to be even distribution of all points in the top-plane. For the case in Figure 4.13 we have occlusion from a box in front of it that results in a missing part in the top-plane. Another occurs when we capture a point cloud from the Zivid-camera. The deeper the points are in the scene, the less dense the number of points are in the neighborhood, which gives us an uneven distribution of points.

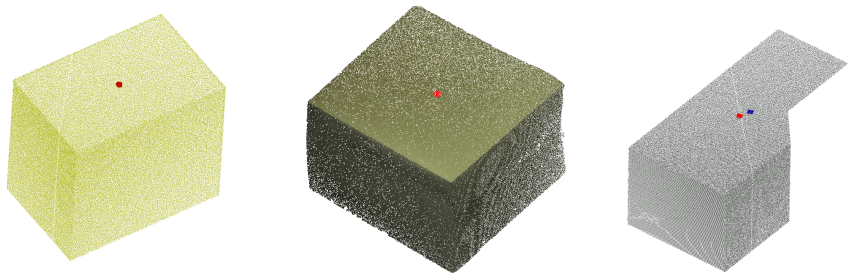


Figure 4.13: Red is the centroid of the top plane and the blue point is the true center for the top-plane.

4.1.5 Box position estimation

In our problem, we have two frames of reference, Camera-frame, and Base-frame, Figure 4.14. The Base-frame is thought of as the known frame by, e.g., the robotic manipulator. The experiment and result done for this part are only done with simulated data. We will look into the result both for describing the position error both in the Base-frame and the Camera-frame.

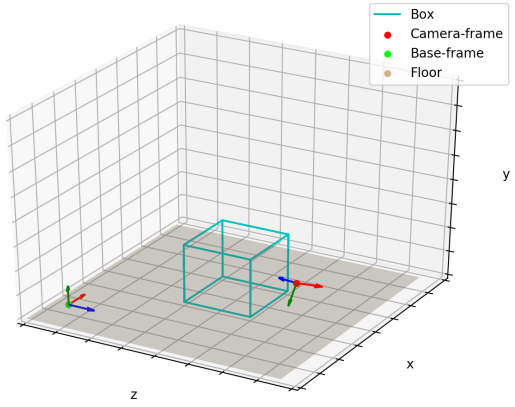


Figure 4.14: Frames of refrence in our problem.

The experiment conducted have the following properties:

- Random placed within the scene.
- Number of point clouds: 50
- Number of boxes in each point cloud: 1
- Total number of boxes: 50.
- Fixed-feature: all others.

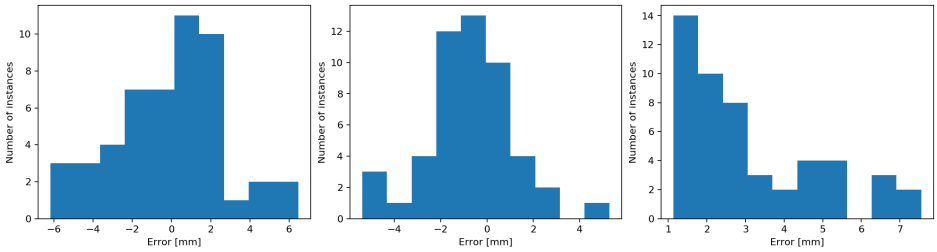
In Table 4.3 we have not included the position in height, since it is already coved in 4.1.4. As we can see from the result of both position in the Base-frame, 4.3, and the Camera-frame, 4.4, have both an accuracy within an 5 mm. These results are more than sufficient for a robotic manipulator to pick up. An interesting observation is that the error in Camera-frame is less than the Base-frame. This has likely to do with that when describing the position from the Base-frame we have to do to estimations. Both the origin of the Base-frame and the position of the box. The result given for the Base-frame will, therefore, be dependent on two errors.

Our method will estimate the correct position if we have a partly occluded top-plane. If we look at the case in Figure 4.13 we have a big part of the top-plane occluded by computing the center of the box as the centroid of all the points. The center will be placed

with a distance of 45.5 mm from the true center. With our method, we get a significant improvement with only 5.8 mm from the true center.

	x-axis	z-axis	Euclidean distance
Mean error $[mm]$	-0.16	-0.62	3.10
Root mean square $[mm]$	2.81	1.96	3.58
Standard deviation error $[mm]$	2.80	1.86	1.78

Table 4.3: Results box position for Base-frame.

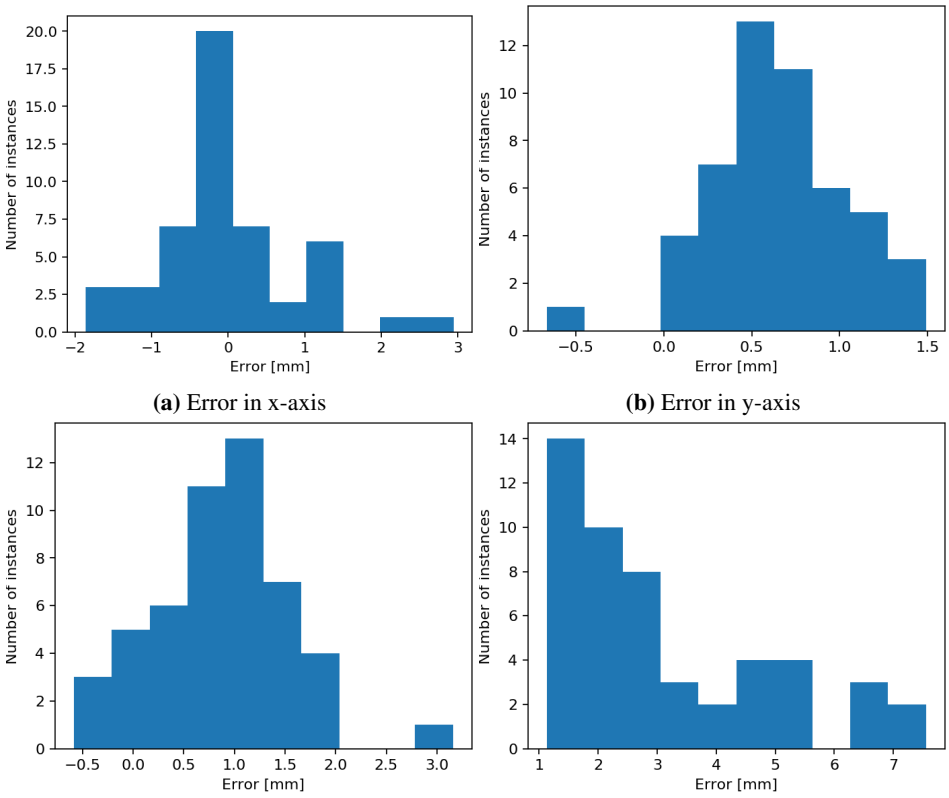


(a) x-axis **(b)** z-axis **(c)** Euclidean distance

Figure 4.15: Histogram of error in calculation of box position in Base-frame.

	x-axis	y-axis	z-axis	Euclidean distance
Mean error $[mm]$	-0.00	0.87	0.65	1.53
Root mean square error $[mm]$	0.75	1.10	0.75	1.63
Standard deviation error $[mm]$	0.93	0.68	0.39	0.55

Table 4.4: Results box position for Camera-frame.



(a) Error in x-axis

(b) Error in y-axis

(c) Error in z-axis

(d) Error in euclidean distance

Figure 4.16: Histogram of error in calculation of box position in Camera-frame.

4.1.6 Box orientation estimation

As well as for the box position, we can describe the orientation in two frames. Since the boxes is supported by the XZ-plane in the Base-frame, we need only to describe the orientation with one angle. For the case of the Camera-frame, there will be three rotations, roll, pitch, yaw. The experiments and results for this part can only be done in the simulator.

The experiment conducted have the following properties:

- Random orientation in about y-axis in Base-frame.
- Number of point clouds: 50
- Number of boxes in each point cloud: 1
- Total number of boxes: 50.
- Fixed-feature: random box size.

As we can see from the Table 4.5 and Figure 4.17 the error is only i bias of $0.49°$. If we look at the mean of the error of the camera orientation for the same instances, we also have a bias of $0.53°$. This is the same case as for the position error where the results contain two errors, which makes sense when we look at the error in roll, pitch, and yaw for the Camera-frame, where the mean is zero and a standard deviation almost equal to zero.

	y-axis
Mean error	$0.49°$
Root mean square error	$0.51°$
Standard deviation error	$0.11°$

Table 4.5: Results box orientation calculation.

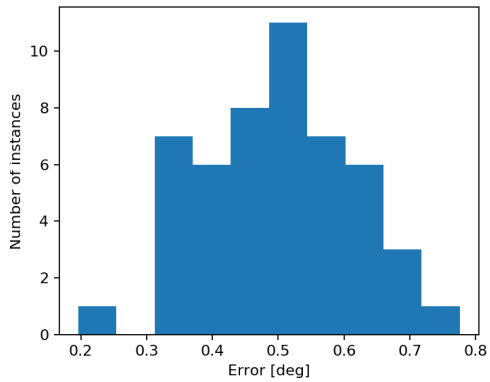


Figure 4.17: Histogram of the orientation error about y-axis in Base-frame.

4.1.7 Box size calculation

An essential feature for the industry is the box size. This is the feature that is the easiest to see as a measurement of cost. For industrial application where they are using fixed-sized cardboard, the size could vary above 2 cm. Since we have measurements of the boxes sizes, we do evaluate our method with the point clouds from the Zivid-camera. However, to get a comprehensive evaluation, we will use the simulator as well with accurate measurements and much more instances.

The experiment conducted with the Zivid-camera has the following properties:

- Two fixed sized boxes
- Number of point clouds: 20
- Number of boxes in each point cloud: 2zero
- Total number of boxes: 40

Moreover, the experiment conducted with the simulator has the following properties:

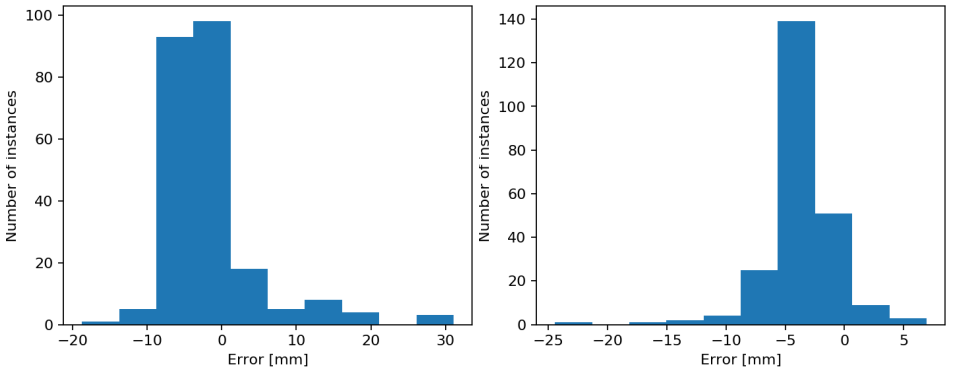
- Random box sizes between 15 cm - 40 cm for each of the dimensions.
- Number of point clouds: 100
- Number of boxes in each point cloud: $1 - 4$
- Total number of boxes: 235.
- Fixed-feature: all others.

In the Table 4.6, Figure 4.18 and 4.19 we can se the results of our experiment. The overall performance of the size calculation is pretty good and is sufficent for a indutrial application where the required accuracy is in cm-range As we can see from Figure 4.18 and 4.19 we have a few cases where we have error above 1 cm.

In the calculation of the width and height, we use the edges calculated in last part, 3.2.2. An assumption we have done is that these edges are parallel to each other. This assumption have we experienced that is not always true. In the case of data from the simulator, we have a deviation of up to 2.75° between the parallel edges. If we look at the instances at the heads and tails of Figure 4.18 all the that have an absolute error above 10 mm have also deviation above 1.0°. For the instances with the must error, above 20 mm, we a deviation above 1.94°.

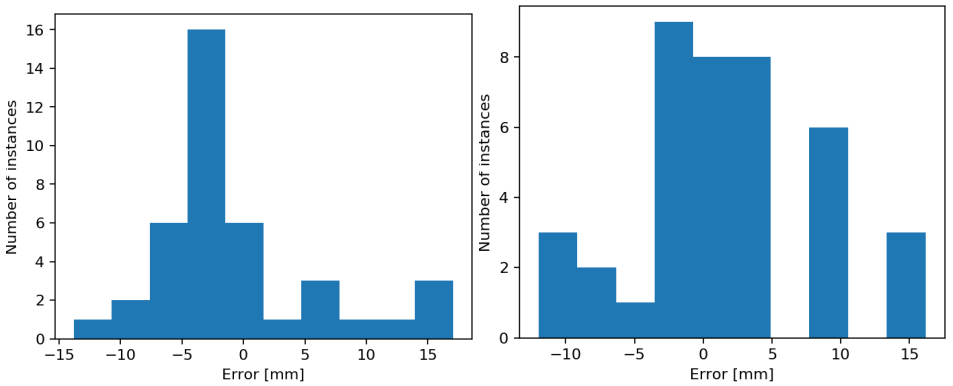
	Zivid		Simulator	
	Width	Length	Width	Length
Mean error $[mm]$	-0.34	-0.09	-1.66	-3.59
Root mean square error $[mm]$	6.92	7.92	6.47	4.68
Standard deviation error $[mm]$	6.98	8.83	6.25	2.99

Table 4.6: Results box size calculation.



(a) Width error **(b)** Length error

Figure 4.18: Histogram of error in calculation of box sizes from simualtor data.



(a) Width error **(b)** Length error

Figure 4.19: Histogram of error in calculation of box sizes from Zivid data.

In search of a method that is more robust, we need a method that is independent of geometric properties between edges. Our first suggestion is first to utilize that the estimation of the orientation has very high performance. This by computing the intersection between the line that is along the orientation of the box and the edges. This line can be described as:

$$\mathbf{l}_c(\alpha) = \mathbf{p}_c \times \left(\mathbf{p}_c + [\sin(\alpha)\ \cos(\alpha)\ 1]^\top\right) \tag{4.1}$$

Where α is the orientation of the box and $\mathbf{p}_c$ is the center of the top-plane. Then the width and length is given by:

$$w = \|\mathbf{l}_{w_1} \times \mathbf{l}_c(\alpha) - \mathbf{l}_{w_2} \times \mathbf{l}_c(\alpha)\| \qquad l = \|\mathbf{l}_{l_1} \times \mathbf{l}_c(\alpha) - \mathbf{l}_{l_2} \times \mathbf{l}_c(\alpha)\| \tag{4.2}$$

Where $\mathbf{l}_{w_1}, \mathbf{l}_{w_2}, \mathbf{l}_{l_1}, \mathbf{l}_{l_2}$ are the lines on the edges for the box.

In this thesis, only ideal simulation data has been used to evalutate the pose, since there is more noise in captured point clouds. Rodriguez-Garavito et al. (2018) suggests both width, length and orientation could be found simoultaneously. They do this discretization of the orientation and measuring the distance, with (4.2), for each step. The features can be extracted where the distance is smallest.

We suggest that we can solve this as an optimization problem:

$$\min_{\alpha} \|\mathbf{l}_{w_1} \times \mathbf{l}_c(\alpha) - \mathbf{l}_{w_2} \times \mathbf{l}_c(\alpha)\| + \|\mathbf{l}_{l_1} \times \mathbf{l}_c(\alpha + \frac{\pi}{2}) - \mathbf{l}_{l_2} \times \mathbf{l}_c(\alpha + \frac{\pi}{2})\|$$
$$\text{subject to} \quad |\alpha| < \frac{\pi}{2}$$

With this optimization problem we will not get an quantization error, and reduce the amount of unnecessarily itterations.

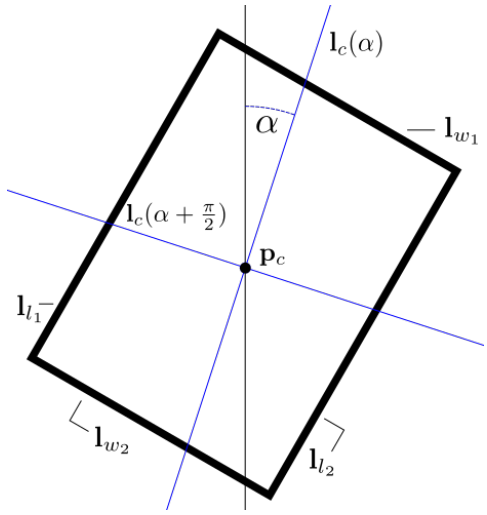


Figure 4.20: Illustration of the optimization to find the width and length.

4.1.8 Run-time

The run-time of the method could be interesting in many cases. The most important is to minimize the idle time that could be caused by the computations. There is a possibility that there is a need for detection the scene multiple time, e.g., if there are disturbances that could cause the boxes to move. Also, the most computation expensive part of the complete system is the Robotic bin packing method. There should be prioritized to run the optimization as long as possible such that it could find any new better solution.

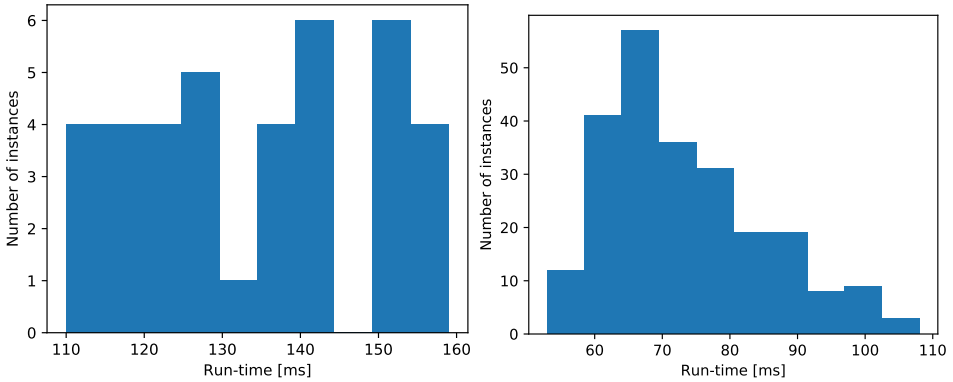
The experiment that is conducted used is the same as for the box size experiment; this dataset has the most number of different boxes. The results could be seen in Table 4.7 and Figure 4.21.

The initial goal given in the limitations was to detect the boxes within one second. As seen from the results beneath that is achieved with the good margin. As there can be seen it is a quite big gap between the point clouds from the Zivid-camera and the simulator. This dataset has the most number of boxes and also with variable size there is a variable number of points for each box. Both of the datasets have boxes sizes in the same range. However, since the simulator has a larger focal length than the Zivid-camera, there will be less dense point cloud for the boxes from the simulator. This results in a computational time that is faster for the simulator.

To get an intuition of what the run-time for this method, it can be compared to the capture time of the Zivid-camera. To capture one point cloud the Zivid-camera uses 100 ms, section 2.1. However, there is also common to use High-dynamic-range, HDR, which captures multiple images at different exposures, then the capture time will be above 200 ms pr. point cloud. For this method, it will be a bit slower than a point cloud captured without HDR, and faster if the point cloud is captured with HDR.

	Zivid	Simulator
Mean run-time $[ms]$	134	73
Standard deviation run-time $[ms]$	15	11

Table 4.7: Results run-time for detection of one box



(a) Zivid **(b)** Simulator

Figure 4.21: Histogram run-time for detection of one box.

4.2 Box packing

In this section, the results of the Robotic bin packing method developed in this thesis will be presented and analyzed. As stated in the introduction to the thesis, 1.2.1, there are some limitations considered when testing this method. RBPP can be stated as a very complex problem, and there exist no trivial solution. Therefore it is important to confine the problem such that it can be feasible for a masters thesis. Most of these restrictions can be made on the complexity that is presented below.

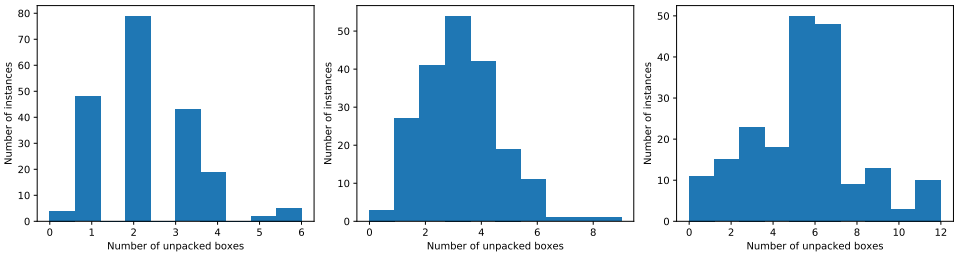
First, two experiments are conducted to investigates performance of two fields; the optimization of the stacking sequence, and the performance of different heuristic methods. Lastly, there will be an experiment on the whole Robotic bin packing method.

In the method that is described in section 3.3, some constants need to be set. While gathered the results for all the experiments that solve RBPP, there have been used:

- Number of boxes on pick, $N_{pick} = 3$
- Number of stacking packing solutions to decide the cost, $N_{cost} = 8$
- Genetic algorithm:
  - Population size: 8
  - Elitism size: 2
  - Generations: 5
  - Probability of being added into the mating pool, $prob_t$: 0.85
  - Probability of crossover, $prob_c$: 0.75
  - Probability of crossover, $prob_m$: 0.5

For solving the BPP problem, there is only used the constants beneath the GA bullet. The constants ”Population size” and ”Generations” are changed to 30 and 10. This is due that the computational time for the RBPP is much higher since it solves multiple BPP for each packing operation.

To compare all the results, all test instances are generated in such a way that there will always be an optimal solution. Also, there is conducted a test on all instances where the boxes are packed with only heuristic packing strategy. With only this heuristic packing strategy, the packing can be thought of as a stupid robotic manipulator. The results for only heuristic packing strategy are presented below in Figure 4.22. As expected, there are many unpacked boxes without any optimization.



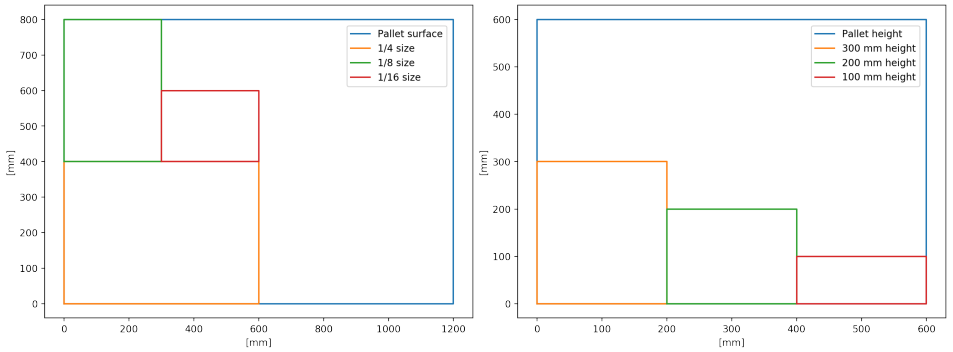
(a) Maximum height 600 mm **(b)** Maximum height 800 mm **(c)** Maximum height 1200 mm

Figure 4.22: Only heuristic packing strategy of boxes, no optimization done

4.2.1 Data

The data considered for this thesis is based on experience from Rocketfarm AS. It is based on a typical industrial application that is fitted for with limitations in this thesis. Solving this type of problems will bring valuable knowledge both as a service and how they can address RBPP.

The data that will be tested with is based on stacking on a pallet. A EURO pallet has a surface area 800 mm x 1200 mm. The limitations that we are given is that the boxes we are going to pack have three different footprints. Where the footprints of the boxes are 1/4, 1/8 and 1/16 of a pallets surface area. These footprints will be $600\text{mm} \times 400\text{mm}$, $300\text{mm} \times 400\text{mm}$ and $300\text{mm} \times 200\text{mm}$. For the height of the boxes, there will be three different height, 300 mm, 200 mm, and 100 mm. A valid packing solution has to fulfill the physical constraints on the pallet. The surface area of the pallet gives the constraints in width and length. While the constraint in height is not trivial since, in industrial application, the stability of the packing solution can vary depending on the objects. Therefore there will be three different maximum heights in the dataset, 600 mm, 800 mm, and 1200 mm. The datasets have all 200 instances in each, and there is one dataset for each height, in total there are 600 instances.



(a) Boxes footprints **(b)** Boxes height

Figure 4.23: Diffrent boxes sizes that are in the dataset

Instances generator

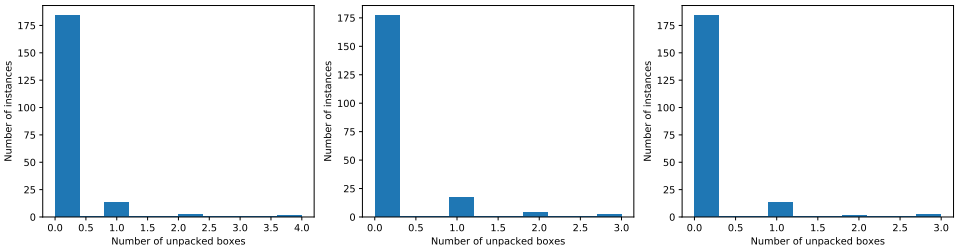
In the lack of any dataset that includes similar types of problems, there has been developed a instances generator. The properties of the instances generator are as follows:

- Scalable dimensions of the pallet.
- Boxes surface area has to be divisible on the pallet surface areas, i.e., 1/4, 1/8, 1/16, 1/32,...
- Boxes heights can be assigned within the maximum height of the packing height.
- Guarantees that there exists a stacking pattern to the problem which has no waste-space if it is solved as a BPP.

The last property is important in our problem. Since all of the instances generated is known to have a solution where all the boxes fit inside. The performance metric can be both number of unpacked boxes and how much –space there is.

4.2.2 Solving the problem as BPP

In order to use the optimization of stacking sequence to decide which box to pick next, the performance of the algorithms must be validated. To do this, it has been conducted an experiment where the instances are solved as a BPP, i.e., possible to change order on all boxes. This experiment is on all three datasets, as described above. The results are presented below



(a) Maximum height 600 mm **(b)** Maximum height 800 mm **(c)** Maximum height 1200 mm

Figure 4.24: Problem solved as an BPP where the order of the boxes can be change

By comparing the result with, Figure 4.24, and without, Figure 4.22, optimization there are major improvements. As these problems are very simplified for a BPP, it should be a requirement that it find an optimal solution. The most important is that it confirms us that it is possible to use this GA in the decision making of the Robotic bin packing.

To further investigate what can cause the error, it has been looked into the case in Figure 4.25. It has three unpacked boxes have sizes 300 mm $\times$ 400 mm $\times$ 100 mm, while the waste-space have footprint 300 mm $\times$ 200 mm. The volume of the waste-space is equal to the space of the unpacked boxes, so there are no fundamental placement errors. The problem could probably have been solved with running the optimization for further generations. Then the unpacked boxes would be placed earlier in a fitted footprint.

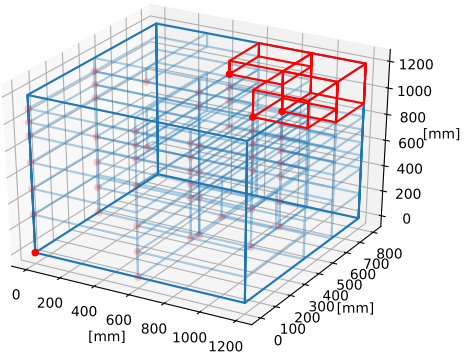
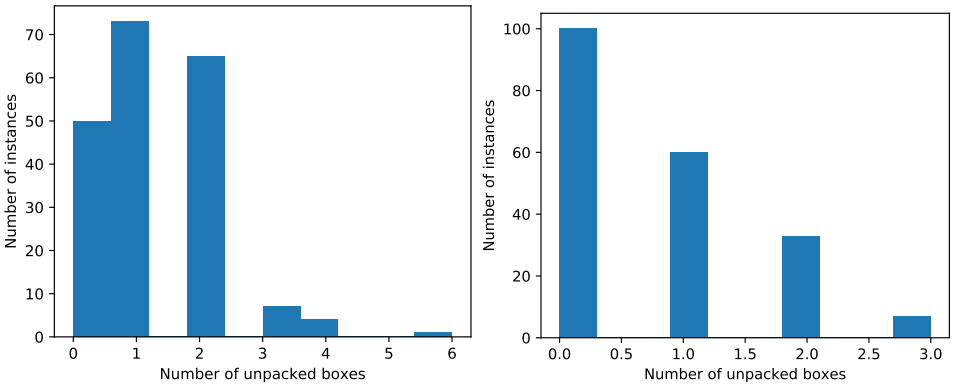


Figure 4.25: Illustration of a solution with 3 unpacked boxes, red boxes are waste-space

4.2.3 Heuristic packing strategy comparison

The most crucial part to get well performance on a Robotic bin packing method is the packing strategy. With the Robotic bin packing method, two different scenarios has been

tested. One with a heuristic packing strategy where the feasible packing space is sorted after the distance from the origin. The other one use a Deep Bottom Left with Fill, DBLF. These are tested on the full robotic bin packing system with the dataset with a height of 600 mm. The results are presented below

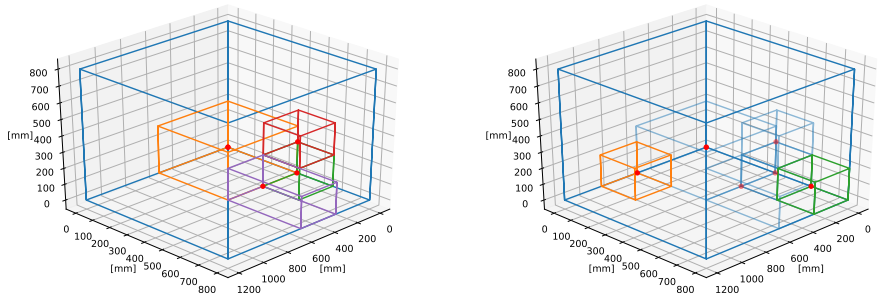


(a) Distance to the origin **(b)** DBLF

Figure 4.26: Comparison of heuristic packing strategy

When investigating the result, a DBLF shows a much better performance. The distance heuristic had 50 instances with perfect stacking, whereas DBLF had 100 instances. This means distance heuristics had a 25% perfect succes rate, whereas DBLF had 50% perfect success rate.

An example if the decisions can be seen in Figure 4.27, where a box of with footprint 300mm $\times$ 200mm should be packed. For the case of distance heuristic, it is as likely place the box in the orange position, since the distance is closer, while the DBFL will fill the space and place the box inside the missing gap and place it in the green position. The green position seems to be the most clever to choose since there is only possible to fit a box with footprint 300mm $\times$ 200mm. In the orange position, there could be placed both of the two other sizes.

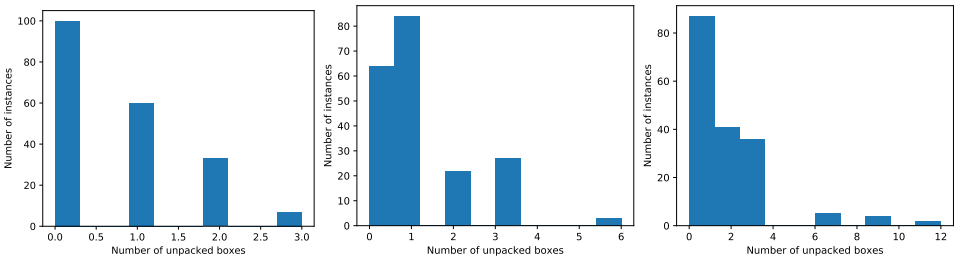


(a) Current packing situation **(b)** Two possible placement

Figure 4.27: Case where the two heuristic can choose differently

4.2.4 Robotic bin packing

Now the final result for the whole method will be presented. These are conducted on all three datasets, i.e., 200 in each dataset with three different heights.



(a) Maximum height 600 mm **(b)** Maximum height 800 mm **(c)** Maximum height 1200 mm

Figure 4.28: Problem solved as an RBPP

The overall performance is seen in Figure 4.28. In 66 % of the cases, its performance as good as optimal or only one box was not packed. As there also could be seen in the figure is that for the case with more boxes, i.e., maximum height of 800 mm and 1200 mm, there are more cases with outliers. In the dataset with a height of 1200 mm, there are 23 cases where more than five boxes have not been packed. This is caused by that when there are more boxes, it will be more opportunities than three large boxes will arrive at the same time. At the end of the packing, the feasible space is getting smaller. If then three large boxes arrive at the same time, it will not have feasible space for any of them and have to discard all three of them. The number for times this occurs can be seen from the figures Figure 4.28b and Figure 4.28c. The instances where there are multiples of 3 unpacked boxes is the most dominant in the figures.

Some of the most interesting results to look at fitness during training. In Figure 4.29 there are the 6 worst instances in the dataset with height 1200, tail of Figure 4.28c. This fitness was the best when the robotic bin packing was choosing which box to pick. For each of the plot, there is possible to count the number of clear steps. The instance with nine unpacked boxes have three steps, and the instance with 12 unpacked boxes have four steps. This is also an observation that the method had not found any feasible space for all the boxes when the decision was made. Another observation that could be made is that there are some parts where the cost suddenly seems to be better. This could occur because the optimization found a possible future solution that it previously did not found.

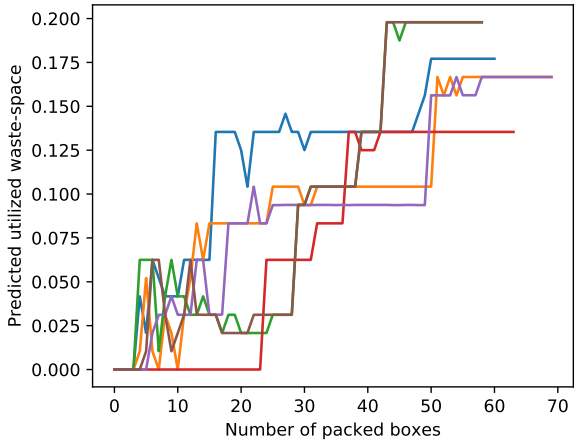


Figure 4.29: The cost of the 6 worst performing instances. The green and brown is the instance with 12 unpacked boxes and the rest have 9 unpacked boxes.

To analyze the results, there will be compared to the results from only heuristic packing, Figure 4.26, and that the dataset is generated to have a solution when solved as a BPP. The pure heuristic packing could be thought of as a worst-case situation if the robotic packing performed worst then that there is no need for any optimization. In Table 4.8, there is a comparison of all the instance. The overall performance is that 91.8% of the instances performed it is better than or equal to the pure heuristic packing strategy.

		Heuristic packing strategy		
Dataset	Optimal	Better	Equal	Worst
600 mm	100	151	33	16
800 mm	64	156	25	19
1200 mm	33	151	35	14

Table 4.8: Results for the robotic bin packing method. The results are in number of instances

Out of 600 cases, the heuristic pacakging strategy managed to find an optimal solution for 8 of them. For 5 of the 8 cases in this subset, the robotic bin packing method had a poorer

performance than the heuristic packaging strategy. This means that the optimization had the oppurtinty to place all the cases optimally, but failed to do so. This gives a failure rate of 5/600.

4.2.5 Comments on the result

At the end of the work on this thesis, a minor error in the implementation of the RBPP was discovered. The error causes the following incorrect behavior: The last three boxes could be placed in the same place, i.e. the last two boxes could be placed at the same place as the third last box. This error will not affect the results in serious deviations in the findings. Due to the fact that the mistake was found just before the deadline of this thesis, it was not possible to correct it and redo all results within time. Due to the relatively mild implications on this bug, the results should still provide a good confirmation of the performance of the methods.

Chapter 5

Conclusion and future work

5.1 Conclusion

Robotic solutions in industrial applications get more important for every year. Repetitive tasks are easy to automate since many of them could be programmed with sequential maneuvers. Palletizing of known sized boxes is a typical example of these tasks. THowever, when palletizing boxes with mixed or even unknown dimensions, new challenges occur. Where should the robot grip, where should the box be placed, and at which order should the robot pick the boxes are crucial questions. These problems can be divided into several subproblems, for instance how to handle damaged boxes and internal stability of pallets due to the order and patterns to palletize. This thesis emphasizes on solving two of the sub problems found in a fully automated palletizing system for mixed size boxes found in industry.

The first suggestion is the Box detection method that utilizes the accuracy and precision of the Zivid-camera to detect boxes. This method use known operations for point clouds to calculate the transformation to a known reference frame, remove constant noise, detects all boxes present in the scene, and describes the pose and size of each box. The size estimation has been tested on data captured from the Zivid-camera. The size of the boxes could be described with an accuracy of 0.66 mm and precision 8.83 mm. In order to also get comprehensive results of the pose, a simulator for point cloud representation of object was developed. As seen from section 4.1.5 and 4.1.6, the pose of the method could be described with high accuracy and high precision.

The second suggestion emphasizes the Robotic Bin Packing Problem, section 2.7.4. Becase RBPP is not well studied, research on Bin Packing Problems have been transformed into the RBPP domain. The method calculates the future cost for each of the available boxes and based on the cost it decides which box to pick. The data used for testing is randomly generated with properties that are commonly found in industrial applications. The

overall performance shows in 91.8% of the instances the decisions that the method takes does equally or better performance than a pure heuristic packing strategy. Despite the fact there are limitations on the data and a minor error in the implementation, the results confirm the method is correct.

In conclusion, the methods developed in this thesis shows satisfactory results which could be used in an industrial application. By combining the detection of boxes from the point cloud, captured from the Zivid-camera, with decisions from the Robotic bin packing method, it could be able to fill a typical missing gap in a fully automated palletizing system.

5.2 Future work

As concluded above, the results show the results show these methods could be used to solve parts of the mixed box palletizing problem. This could motivate to further study the problem.

During this thesis, there have been no experiments on a complete system. If it should be taken into consideration to continue with this study, an experiment on a complete system should be conducted. By doing this, a framework for the total system needs to be developed, which makes it easier to test future improvements.

The least robust part of the box detection method is the clustering. In a noisy environment and more uncertainties on the boxes, it needs to be very robust. At the same time, many of the ordinary clustering techniques are time-consuming and therefore, unsuitable for this application. The suggested improvement is to utilize that the Zivid-camera outputs an organized point cloud, and therefore, do segmentation of the boxes in the RGB-image. Then extract the point cloud of the box with the help from the mask it outputs. The segmentation can suggestively be done by a Mask-RCNN trained to segmentate boxes.

The Robotic bin packing is the method with most limitations, and therefore, many improvements could be made. As a start, the instance generator could be made more advanced by adding more variations in sizes. A suggestion to this could be to add a standard deviation on each of the dimensions. Another could be to add continues heights. By doing this it could be interesting to see if there is an advantage to do rotations of the boxes.

In Bin packing problem, the heuristic packing strategy is not that important since there could be done many decisions on the order of the boxes. For the Robotic bin packing problem, there need to be done much more short term decisions that affect the whole solution. Therefore it should be investigated more advanced methods that do not place the boxes on rule-based methods. These could be a packing strategy that does optimization purely on the packing.

The most promising new research on combinatorial optimization is done with deep reinforcement learning. As mention in section 2.7.3, it has been tested with excellent results for Bin packing problems. There would be of big interest to try some of these methods on the Robotic bin packing problem. However, to implement this, there is a need for a large amount of training data, which could be hard to get.

Bibliography

Baker, B. S., Coffman, Jr, E. G., Rivest, R. L., 1980. Orthogonal packings in two dimensions. SIAM Journal on computing 9 (4), 846–855.

Ballard, D. H., 1981. Generalizing the hough transform to detect arbitrary shapes. Pattern recognition 13 (2), 111–122.

Barber, C. B., Dobkin, D. P., Dobkin, D. P., Huhdanpaa, H., 1996. The quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software (TOMS) 22 (4), 469–483.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., Bengio, S., 2016. Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.

Bortfeldt, A., Wäscher, G., 2013. Constraints in container loading–a state-of-the-art review. European Journal of Operational Research 229 (1), 1–20.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., 2009. Introduction to algorithms. MIT press.

Crainic, T. G., Perboli, G., Tadei, R., 2009. Ts2pack: A two-level tabu search for the three-dimensional bin packing problem. European Journal of Operational Research 195 (3), 744–760.

Egeland, O., Gravdahl, J. T., 2002. Modeling and simulation for automatic control. Vol. 76. Marine Cybernetics Trondheim, Norway.

Faroe, O., Pisinger, D., Zachariasen, M., 2003. Guided local search for the three-dimensional bin-packing problem. Informs journal on computing 15 (3), 267–283.

Fekete, S. P., Schepers, J., 1997. A new exact algorithm for general orthogonal d-dimensional knapsack problems. In: European Symposium on Algorithms. Springer, pp. 144–156.

Fekete, S. P., Schepers, J., 2004. A combinatorial characterization of higher-dimensional orthogonal packing. Mathematics of Operations Research 29 (2), 353–368.

Fischler, M. A., Bolles, R. C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24 (6), 381–395.

Garey, M. R., Johnson, D. S., 2002. Computers and intractability. Vol. 29. wh freeman New York.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., Marín-Jiménez, M. J., 2014. Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition 47 (6), 2280–2292.

Gonçalves, J. F., Resende, M. G., 2013. A biased random key genetic algorithm for 2d and 3d bin packing problems. International Journal of Production Economics 145 (2), 500–510.

Hartley, R., Zisserman, A., 2003. Multiple view geometry in computer vision. Cambridge university press.

Hopper, E., 2000. Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods. Ph.D. thesis, University of Wales. Cardiff.

Hu, H., Duan, L., Zhang, X., Xu, Y., Wei, J., 2018. A multi-task selected learning approach for solving new type 3d bin packing problem. CoRR abs/1804.06896.
URL http://arxiv.org/abs/1804.06896

Hu, H., Zhang, X., Yan, X., Wang, L., Xu, Y., 2017. Solving a new 3d bin packing problem with deep reinforcement learning method. arXiv preprint arXiv:1708.05930.

Illingworth, J., Kittler, J., 1988. A survey of the hough transform. Computer vision, graphics, and image processing 44 (1), 87–116.

Kang, K., Moon, I., Wang, H., 2012. A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem. Applied Mathematics and Computation 219 (3), 1287–1299.

Karabulut, K., İnceoğlu, M. M., 2004. A hybrid genetic algorithm for packing in 3d with deepest bottom left with fill method. In: International Conference on Advances in Information Systems. Springer, pp. 441–450.

Lai, K., Chan, J. W., 1997. Developing a simulated annealing algorithm for the cutting stock problem. Computers & industrial engineering 32 (1), 115–127.

Laterre, A., Fu, Y., Jabri, M. K., Cohen, A., Kas, D., Hajjar, K., Dahl, T. S., Kerkeni, A., Beguir, K., 2018. Ranked reward: Enabling self-play reinforcement learning for combinatorial optimization. CoRR abs/1807.01672.
URL http://arxiv.org/abs/1807.01672

Li, X., Zhao, Z., Zhang, K., 2014. A genetic algorithm for the three-dimensional bin packing problem with heterogeneous bins. In: IIE Annual Conference. Proceedings. Institute of Industrial and Systems Engineers (IISE), p. 2039.

Lodi, A., Martello, S., Vigo, D., 1999. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. INFORMS Journal on Computing 11 (4), 345–357.

Lodi, A., Martello, S., Vigo, D., 2002. Heuristic algorithms for the three-dimensional bin packing problem. European Journal of Operational Research 141 (2), 410–420.

Martello, S., Pisinger, D., Vigo, D., 2000. The three-dimensional bin packing problem. Operations research 48 (2), 256–267.

Martello, S., Pisinger, D., Vigo, D., Boef, E. D., Korst, J., 2007. Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem. ACM Transactions on Mathematical Software (TOMS) 33 (1), 7.

Rodriguez-Garavito, C., Camacho-Munoz, G., Álvarez-Martínez, D., Cardenas, K. V., Rojas, D. M., Grimaldos, A., 2018. 3d object pose estimation for robotic packing applications. In: Workshop on Engineering Applications. Springer, pp. 453–463.

Scheithauer, G., 1991. A three-dimensional bin packing algorithm. Elektronische Informationsverarbeitung und Kybernetik 27 (5/6), 263–271.

Schnabel, R., Wahl, R., Klein, R., 2007. Efficient ransac for point-cloud shape detection. In: Computer graphics forum. Vol. 26. Wiley Online Library, pp. 214–226.

Trevor, A. J., Gedikli, S., Rusu, R. B., Christensen, H. I., 2013. Efficient organized point cloud segmentation with connected components. Semantic Perception Mapping and Exploration (SPME).

Vasdal, E., 2018. Box detection in a point cloud. TTK4550 - Engineering Cybernetics, Specialization Project.

Vinyals, O., Fortunato, M., Jaitly, N., 2015. Pointer networks. In: Advances in Neural Information Processing Systems. pp. 2692–2700.

Wang, H., Yanjie Chen, Sep. 2010. A hybrid genetic algorithm for 3d bin packing problems. In: 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA). pp. 703–707.

Wäscher, G., Haußner, H., Schumann, H., 2007. An improved typology of cutting and packing problems. European journal of operational research 183 (3), 1109–1130.

Appendices

Appendix A

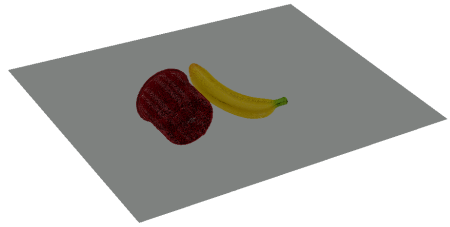
3D point cloud simulator

The 3D point cloud simulatour is made becuse of there are no accurate method to obtain pose data when using capturing point clouds with a 3D-camera. Also it gives the abilety to easly to experiments under controlled environment with a lot of point clouds. The simulator inputs an point cloud generated from CAD-model or mesh-model, and a outputs a orgainzed point cloud with all transformation done to it. The simulator works in four steps:

1. Transformation: Each of the object need to be given a transformation matrix and the scene will also be given one. These transformation matrices are applyes to the respectively point clouds.
2. Projection: First each of the point clouds are projected into the normalized camera-frame. Then the camara matrix is used to find the pixel coordinate for each of the points. If multiple points have same pixel coordinate it will only keep the point closes to the camera, smallest depth-value.
3. Depth-map: For each of the object it is made a depth-map.
4. Merge object: The depth-maps are used as mask to find which of the objects that occludes the others. These masks can then be used to remove occluded points. The result is a orgainzed point cloud.

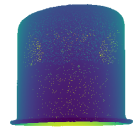
In addition to give out a orgainzed point cloud there the masks could be outputted. These masks can be used for training and testing of segmentation neural networks.

(a) Input point cloud

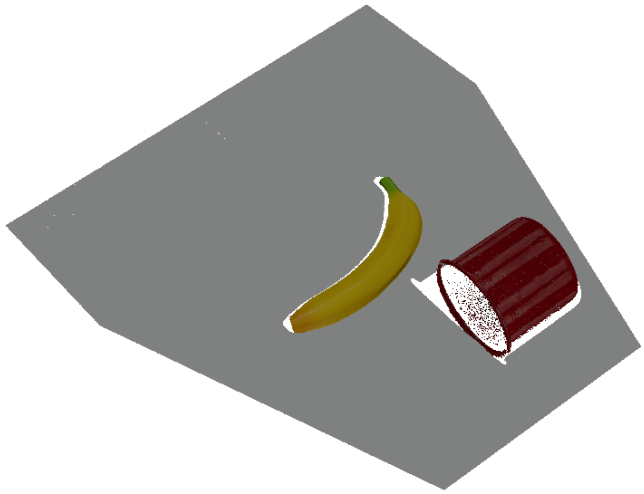


(b) Transformed into the scene

(c) Point cloud projected into the camera, removed occluded points



(d) Depth-map of point cloud



(e) Output point cloud

Figure A.1: An example of the four steps, described above, in the simulator. The camera is placed to look at the sceen with an angle of 45°.

Appendix B

Results Box detection

	Ground truth		Estimated		Error	
#	y-axis	k-axis	y-axis	k-axis	y-axis	k-axis
1	58.0	−26.0	58.0	−25.9	−0.1	0.0
2	41.0	−14.0	41.0	−13.93	−0.07	0.0
3	56.0	32.0	56.0	31.78	0.21	0.0
4	66.0	−29.0	66.0	−28.24	−0.76	0.0
5	48.0	2.0	48.0	0.93	1.07	0.0
6	45.0	−37.0	45.0	−36.29	−0.71	0.0
7	61.0	3.0	61.0	2.52	0.48	0.0
8	47.0	39.0	47.0	38.07	0.93	0.0
9	63.0	5.0	63.0	4.15	0.85	−0.0
10	40.0	29.0	40.0	29.0	0.0	0.0
11	38.0	−5.0	38.0	−4.18	−0.82	0.0
12	56.0	−31.0	56.0	−30.94	−0.06	0.0
13	67.0	0.0	67.0	0.01	−0.01	0.0
14	53.0	−32.0	53.0	−31.81	−0.19	0.0
15	64.0	−37.0	64.0	−36.27	−0.73	−0.0
16	41.0	−2.0	41.0	0.0	−2.0	0.0
17	35.0	39.0	35.0	0.0	39.0	0.0
18	39.0	−7.0	39.0	0.0	−7.0	0.0
19	62.0	29.0	62.0	28.74	0.26	0.0
20	53.0	−39.0	53.0	−38.34	−0.66	0.0
21	37.0	−29.0	37.0	0.0	−29.0	0.0
22	50.0	−15.0	50.0	−14.76	−0.24	0.0
23	40.0	−33.0	40.0	−32.24	−0.76	0.0
24	48.0	9.0	48.0	7.32	1.68	0.0
25	35.0	−24.0	35.0	0.0	−24.0	0.0

Continue on next page

Table B.1 – continued from previous page

	Ground truth		Estimated		Error	
#	y-axis	k-axis	y-axis	k-axis	y-axis	k-axis
26	69.0	−40.0	69.0	−39.47	−0.53	0.0
27	64.0	−16.0	64.0	−15.47	−0.53	0.0
28	56.0	−18.0	56.0	−17.17	−0.83	0.0
29	59.0	26.0	59.0	25.69	0.31	0.0
30	64.0	8.0	64.0	7.8	0.2	0.0
31	43.0	1.0	43.0	0.97	0.03	0.0
32	48.0	−21.0	48.0	−20.77	−0.23	0.0
33	59.0	−13.0	59.0	−12.68	−0.32	0.0
34	52.0	8.0	52.0	7.68	0.32	0.0
35	56.0	37.0	56.0	0.0	37.0	0.0
36	54.0	28.0	54.0	27.67	0.33	0.0
37	58.0	−22.0	58.0	−21.6	−0.4	0.0
38	46.0	−32.0	46.0	−31.99	−0.01	0.0
39	67.0	8.0	67.0	7.91	0.09	0.0
40	52.0	−31.0	52.0	−30.66	−0.34	0.0
41	45.0	33.0	45.0	32.44	0.56	0.0
42	57.0	31.0	57.0	29.72	1.28	0.0
43	67.0	−16.0	67.0	−15.96	−0.04	0.0
44	53.0	11.0	53.0	10.97	0.03	0.0
45	44.0	−36.0	44.0	−36.0	0.0	0.0
46	35.0	17.0	35.0	0.0	17.0	0.0
47	45.0	39.0	45.0	37.65	1.35	0.0
48	63.0	−11.0	63.0	−10.91	−0.09	0.0
49	38.0	9.0	38.0	8.33	0.67	0.0
50	39.0	34.0	39.0	33.94	0.06	0.0

Table B.1: Results camera orientation estimation, all dimensions in degrees.

	Ground truth			Estimated			Error		
#	x-axis	y-axis	z-axis	x-axis	y-axis	z-axis	x-axis	y-axis	z-axis
1	90.0	250.0	800.0	90.66	249.04	800.15	−0.66	0.96	1.16
2	−100.0	250.0	920.0	−93.82	249.14	921.18	−6.18	0.87	6.24
3	230.0	250.0	320.0	229.6	248.74	321.35	0.4	1.26	1.33
4	−50.0	250.0	500.0	−48.02	248.97	500.12	−1.98	1.03	2.23
5	−60.0	250.0	820.0	−57.94	249.07	820.62	−2.06	0.93	2.26
6	−400.0	250.0	750.0	−401.74	248.68	750.09	1.74	1.32	2.18
7	80.0	250.0	510.0	83.61	248.85	508.85	−3.61	1.15	3.78
8	−500.0	250.0	900.0	−501.86	249.12	900.75	1.86	0.88	2.06
9	210.0	250.0	310.0	209.52	249.01	310.67	0.48	0.99	1.1
10	−510.0	250.0	660.0	−507.61	249.02	662.04	−2.39	0.98	2.58
11	−440.0	250.0	870.0	−441.5	248.99	869.86	1.5	1.01	1.81
12	−70.0	250.0	740.0	−76.46	249.1	739.64	6.46	0.9	6.53
13	560.0	250.0	580.0	564.98	249.09	574.73	−4.98	0.91	5.06
14	−200.0	250.0	690.0	−205.19	248.96	689.09	5.19	1.03	5.29
15	160.0	250.0	500.0	161.03	249.0	500.48	−1.03	1.0	1.43
16	440.0	250.0	330.0	437.39	248.95	334.33	2.61	1.05	2.82
17	380.0	250.0	500.0	376.6	248.85	505.38	3.4	1.15	3.59
18	0.0	250.0	980.0	0.81	249.27	979.52	−0.81	0.73	1.09
19	−330.0	250.0	720.0	−334.53	248.79	718.5	4.53	1.21	4.69
20	440.0	250.0	230.0	439.76	249.02	230.5	0.24	0.99	1.01
21	0.0	250.0	690.0	−1.1	249.03	690.78	1.1	0.97	1.47
22	−60.0	250.0	740.0	−60.76	249.04	739.79	0.76	0.96	1.22
23	60.0	250.0	690.0	65.27	248.76	689.5	−5.27	1.24	5.41
24	210.0	250.0	770.0	209.0	248.95	771.4	1.0	1.05	1.45
25	480.0	250.0	610.0	479.33	248.89	611.15	0.67	1.11	1.3
26	180.0	250.0	600.0	180.41	248.84	599.42	−0.41	1.16	1.23
27	−490.0	250.0	460.0	−489.8	249.22	459.04	−0.2	0.78	0.81
28	−560.0	250.0	490.0	−555.77	248.81	491.38	−4.23	1.19	4.4
29	−570.0	250.0	440.0	−570.71	248.77	437.4	0.71	1.23	1.42
30	−410.0	250.0	70.0	−408.3	248.99	72.25	−1.7	1.01	1.98
31	−450.0	250.0	270.0	−451.26	248.97	267.71	1.26	1.03	1.62
32	−200.0	250.0	720.0	−196.88	249.33	721.12	−3.12	0.67	3.19
33	−390.0	250.0	430.0	−386.6	248.97	432.93	−3.4	1.03	3.56
34	−120.0	250.0	400.0	−119.14	249.15	400.59	−0.86	0.85	1.21
35	320.0	250.0	380.0	318.49	248.48	382.21	1.51	1.52	2.14
36	150.0	250.0	250.0	151.52	249.02	249.36	−1.52	0.98	1.8
37	170.0	250.0	300.0	168.22	249.01	301.56	1.78	0.99	2.04
38	460.0	250.0	960.0	462.22	249.05	959.58	−2.22	0.95	2.42
39	180.0	250.0	890.0	182.15	249.35	890.44	−2.15	0.65	2.25
40	470.0	250.0	640.0	473.9	249.13	638.33	−3.9	0.87	4.0
41	−250.0	250.0	250.0	−248.0	248.92	254.04	−2.0	1.08	2.28
42	−190.0	250.0	460.0	−190.75	248.97	460.05	0.75	1.03	1.27

Continue on next page

Table B.2 – continued from previous page

	Ground truth			Estimated			Error		
#	x-axis	y-axis	z-axis	x-axis	y-axis	z-axis	x-axis	y-axis	z-axis
43	−460.0	250.0	370.0	−459.64	248.94	370.17	−0.36	1.06	1.12
44	200.0	250.0	440.0	198.59	249.06	441.52	1.41	0.94	1.7
45	440.0	250.0	690.0	433.98	248.95	694.43	6.02	1.05	6.12
46	−450.0	250.0	30.0	−452.09	249.17	31.88	2.09	0.83	2.25
47	580.0	250.0	570.0	578.0	249.24	572.02	2.0	0.75	2.13
48	460.0	250.0	460.0	458.0	249.05	461.49	2.0	0.95	2.22
49	−390.0	250.0	840.0	−385.39	249.04	842.88	−4.61	0.96	4.7
50	−600.0	250.0	200.0	−600.31	248.76	198.98	0.31	1.24	1.28

Table B.2: Results position estimation, all dimensions in mm.

	Ground truth	Estimated	Error
1	35.0	34.59	0.41
2	33.0	32.49	0.51
3	−14.0	−14.56	0.56
4	−32.0	−32.56	0.56
5	−4.0	−4.53	0.53
6	1.0	0.49	0.51
7	−6.0	−6.62	0.62
8	−10.0	−10.4	0.4
9	−16.0	−16.45	0.45
10	31.0	30.43	0.57
11	−3.0	−3.49	0.49
12	6.0	5.5	0.5
13	−34.0	−34.49	0.49
14	−25.0	−25.48	0.48
15	28.0	27.42	0.58
16	−39.0	−39.47	0.47
17	20.0	19.47	0.53
18	−35.0	−35.52	0.52
19	−7.0	−7.59	0.59
20	14.0	13.43	0.57
21	−21.0	−21.43	0.43
22	4.0	3.5	0.5
23	−24.0	−24.56	0.56
24	10.0	9.41	0.59
25	−12.0	−12.58	0.58
26	−16.0	−16.44	0.44
27	−8.0	−8.52	0.52
28	16.0	15.47	0.53
29	−40.0	−40.63	0.63
30	23.0	22.44	0.56
31	−34.0	−34.53	0.53
32	−35.0	−35.46	0.46
33	38.0	37.44	0.56
34	−31.0	−31.52	0.52
35	−13.0	−13.54	0.54
36	−34.0	−34.57	0.57
37	37.0	36.49	0.51
38	−6.0	−6.5	0.5
39	26.0	25.61	0.39
40	6.0	5.46	0.54
41	−7.0	−7.45	0.45
42	−32.0	−32.69	0.69
43	−1.0	−1.57	0.57

Continue on next page

Table B.3 – continued from previous page

	Ground truth	Estimated	Error
44	−34.0	−34.53	0.53
45	−36.0	−36.52	0.52
46	21.0	20.44	0.56
47	−24.0	−24.54	0.54
48	35.0	34.44	0.56
49	−40.0	−40.57	0.57
50	−18.0	−18.56	0.56

Table B.3: Results orientation estimation relative to Base-Frame, all dimensions in degrees.

	Ground truth			Estimated			Error		
#	Width	Height	Lenght	Width	Height	Lenght	Width	Height	Lenght
1	205.0	165.0	230.0	212.34	165.0	234.91	−7.34	0.0	−4.91
2	216.0	155.0	205.0	202.33	152.61	204.28	13.67	2.39	0.72
3	205.0	165.0	230.0	218.79	170.3	239.11	−13.79	−5.3	−9.11
4	205.0	155.0	216.0	206.19	158.6	200.39	−1.19	−3.6	15.61
5	205.0	165.0	230.0	213.42	165.68	241.97	−8.42	−0.68	−11.97
6	216.0	155.0	205.0	207.3	155.55	207.22	8.7	−0.55	−2.22
7	205.0	165.0	230.0	214.22	167.04	241.61	−9.22	−2.04	−11.61
8	216.0	155.0	205.0	200.42	157.36	207.96	15.58	−2.36	−2.96
9	205.0	165.0	230.0	209.78	162.54	239.57	−4.78	2.46	−9.57
10	205.0	155.0	216.0	206.98	151.6	208.14	−1.98	3.4	7.86
11	165.0	230.0	205.0	167.72	228.78	196.49	−2.72	1.22	8.51
12	205.0	155.0	216.0	207.61	154.2	208.13	−2.61	0.8	7.87
13	230.0	165.0	205.0	212.96	154.59	207.93	17.04	10.41	−2.93
14	216.0	155.0	205.0	212.96	154.59	207.93	3.04	0.41	−2.93
15	165.0	205.0	230.0	169.41	201.73	233.41	−4.41	3.27	−3.41
16	205.0	155.0	216.0	205.98	152.81	207.76	−0.98	2.19	8.24
17	165.0	205.0	230.0	166.6	206.45	233.2	−1.6	−1.45	−3.2
18	155.0	205.0	216.0	153.42	204.09	214.72	1.58	0.91	1.28
19	165.0	205.0	230.0	167.07	204.14	238.87	−2.07	0.86	−8.87
20	205.0	155.0	216.0	205.56	155.08	199.84	−0.56	−0.08	16.16
21	230.0	205.0	165.0	235.33	207.0	163.45	−5.33	−2.0	1.55
22	205.0	155.0	216.0	209.89	156.39	240.86	−4.89	−1.39	−24.86
23	230.0	205.0	165.0	231.42	204.78	166.64	−1.43	0.22	−1.63
24	216.0	155.0	205.0	209.26	154.96	204.82	6.74	0.04	0.18
25	230.0	205.0	165.0	232.22	204.48	163.67	−2.22	0.52	1.33
26	216.0	155.0	205.0	209.77	152.44	202.39	6.23	2.56	2.61
27	230.0	205.0	165.0	233.07	205.2	165.34	−3.07	−0.2	−0.34
28	205.0	155.0	216.0	207.48	154.73	201.59	−2.48	0.27	14.41
29	230.0	165.0	205.0	233.32	164.64	200.42	−3.31	0.36	4.58
30	155.0	205.0	216.0	161.91	202.75	213.6	−6.91	2.25	2.4
31	230.0	165.0	205.0	233.32	164.64	200.42	−3.31	0.36	4.58
32	155.0	205.0	216.0	161.91	202.75	213.6	−6.91	2.25	2.4
33	230.0	165.0	205.0	233.32	164.64	200.42	−3.31	0.36	4.58
34	155.0	205.0	216.0	161.91	202.75	213.6	−6.91	2.25	2.4
35	230.0	165.0	205.0	232.8	165.44	205.56	−2.8	−0.44	−0.56
36	216.0	155.0	205.0	199.23	156.47	208.38	16.77	−1.47	−3.38
37	230.0	165.0	205.0	232.72	159.41	204.09	−2.72	5.59	0.91
38	155.0	216.0	205.0	157.24	210.61	195.16	−2.24	5.39	9.84
39	205.0	165.0	230.0	205.88	164.7	233.32	−0.88	0.3	−3.32
40	155.0	216.0	205.0	156.53	214.41	195.55	−1.53	1.59	9.45

Table B.4: Results size estimation on Zivid-data, all dimensions in mm.

	Ground truth			Estimated			Error		
#	Width	Height	Lenght	Width	Height	Lenght	Width	Height	Lenght
1	370.0	290.0	250.0	375.03	289.07	253.75	−5.03	0.93	−3.75
2	200.0	260.0	270.0	204.76	258.94	269.53	−4.76	1.06	0.47
3	150.0	360.0	180.0	151.09	359.25	183.32	−1.09	0.75	−3.32
4	170.0	390.0	380.0	172.64	389.34	383.57	−2.65	0.66	−3.57
5	220.0	300.0	250.0	224.97	298.92	252.72	−4.97	1.08	−2.72
6	380.0	320.0	200.0	384.84	318.92	202.42	−4.84	1.08	−2.42
7	270.0	390.0	320.0	260.56	388.85	321.61	9.44	1.15	−1.61
8	260.0	270.0	240.0	251.5	268.89	245.88	8.5	1.11	−5.88
9	300.0	200.0	320.0	286.83	198.84	322.74	13.17	1.16	−2.74
10	290.0	170.0	280.0	295.08	169.12	283.12	−5.08	0.88	−3.12
11	160.0	160.0	220.0	162.98	158.82	223.44	−2.98	1.18	−3.44
12	360.0	330.0	300.0	363.42	329.34	302.41	−3.42	0.66	−2.41
13	200.0	290.0	230.0	197.32	289.06	242.07	2.68	0.94	−12.07
14	260.0	290.0	200.0	267.97	289.03	203.77	−7.97	0.97	−3.77
15	260.0	360.0	360.0	264.89	358.97	362.95	−4.89	1.03	−2.95
16	350.0	300.0	260.0	332.25	299.22	263.59	17.75	0.78	−3.59
17	340.0	170.0	180.0	343.59	169.14	184.36	−3.59	0.86	−4.36
18	350.0	330.0	390.0	353.62	329.38	393.63	−3.62	0.62	−3.63
19	380.0	230.0	350.0	381.75	229.11	354.23	−1.75	0.89	−4.23
20	280.0	280.0	220.0	283.09	279.06	225.02	−3.09	0.94	−5.02
21	260.0	340.0	180.0	254.26	338.78	177.25	5.74	1.22	2.75
22	380.0	380.0	310.0	386.68	378.97	313.55	−6.68	1.03	−3.55
23	380.0	250.0	220.0	377.82	248.88	222.96	2.18	1.12	−2.96
24	180.0	370.0	250.0	184.28	369.0	252.2	−4.28	1.0	−2.2
25	270.0	320.0	170.0	269.65	318.99	173.6	0.35	1.01	−3.6
26	370.0	290.0	360.0	388.74	289.32	363.21	−18.74	0.68	−3.21
27	160.0	170.0	160.0	194.89	348.99	302.59	−4.63	1.01	−3.72
28	190.0	350.0	300.0	164.63	168.99	163.72	−4.89	1.01	−2.59
29	210.0	240.0	370.0	207.46	239.01	375.79	2.53	0.99	−5.79
30	230.0	250.0	240.0	234.45	249.03	243.93	−4.45	0.97	−3.93
31	230.0	240.0	320.0	235.05	239.13	324.43	−5.05	0.87	−4.43
32	260.0	170.0	190.0	259.24	168.82	192.8	0.76	1.18	−2.8
33	270.0	310.0	250.0	259.41	308.92	255.3	10.59	1.08	−5.3
34	230.0	200.0	150.0	235.84	199.18	150.34	−5.84	0.82	−0.34
35	190.0	270.0	160.0	171.98	269.03	165.72	18.02	0.97	−5.72
36	230.0	260.0	190.0	234.24	259.4	187.0	−4.24	0.6	3.0
37	340.0	390.0	150.0	312.58	388.94	151.91	27.42	1.06	−1.91
38	370.0	150.0	160.0	373.68	149.02	162.33	−3.68	0.98	−2.33
39	330.0	210.0	230.0	334.64	209.2	233.28	−4.64	0.8	−3.28
40	210.0	260.0	290.0	210.74	258.9	293.09	−0.74	1.1	−3.09
41	160.0	200.0	210.0	163.06	198.98	204.19	−3.06	1.02	5.81
42	160.0	300.0	390.0	163.63	298.88	394.94	−3.63	1.12	−4.94

Continue on next page

Table B.5 – continued from previous page

	Ground truth			Estimated			Error		
#	Width	Height	Lenght	Width	Height	Lenght	Width	Height	Lenght
43	330.0	360.0	150.0	329.65	358.83	153.27	0.35	1.17	−3.27
44	280.0	260.0	260.0	284.29	259.35	262.35	−4.29	0.65	−2.35
45	280.0	340.0	180.0	266.77	339.05	183.68	13.23	0.95	−3.68
46	190.0	360.0	300.0	195.31	359.03	304.68	−5.31	0.97	−4.68
47	390.0	360.0	300.0	393.12	359.12	304.17	−3.12	0.88	−4.17
48	210.0	300.0	250.0	214.76	299.14	246.75	−4.76	0.86	3.25
49	360.0	270.0	180.0	328.92	269.08	181.36	31.08	0.92	−1.36
50	390.0	280.0	200.0	395.58	279.09	200.68	−5.58	0.91	−0.68
51	240.0	310.0	380.0	244.73	309.05	384.03	−4.73	0.95	−4.03
52	190.0	240.0	170.0	194.78	238.95	194.44	−4.78	1.05	−24.44
53	300.0	360.0	340.0	304.62	358.75	344.23	−4.62	1.25	−4.23
54	160.0	170.0	270.0	165.92	169.08	275.04	−5.92	0.92	−5.04
55	170.0	310.0	290.0	175.72	309.12	296.37	−5.72	0.88	−6.37
56	260.0	380.0	380.0	264.55	379.01	382.2	−4.55	0.99	−2.2
57	330.0	150.0	160.0	273.81	369.05	312.33	18.47	1.21	−7.74
58	270.0	370.0	310.0	311.53	148.79	167.74	−3.81	0.95	−2.33
59	280.0	240.0	190.0	223.34	298.91	243.46	13.58	0.85	−3.13
60	220.0	300.0	240.0	266.42	239.15	193.13	−3.34	1.09	−3.46
61	270.0	270.0	390.0	267.82	268.98	393.08	2.18	1.02	−3.08
62	330.0	240.0	160.0	329.97	238.96	164.52	0.03	1.04	−4.52
63	210.0	340.0	200.0	206.05	338.9	204.24	3.95	1.1	−4.24
64	260.0	150.0	260.0	261.09	149.2	265.16	−1.09	0.8	−5.16
65	210.0	210.0	290.0	209.2	209.0	299.53	0.8	1.0	−9.53
66	300.0	320.0	210.0	303.59	319.02	210.04	−3.59	0.98	−0.04
67	360.0	390.0	380.0	343.6	389.02	386.28	16.39	0.98	−6.27
68	350.0	230.0	210.0	354.15	229.01	216.45	−4.15	0.99	−6.45
69	180.0	240.0	260.0	190.1	239.19	258.65	−10.1	0.81	1.35
70	210.0	350.0	250.0	212.24	349.06	254.63	−2.24	0.94	−4.63
71	180.0	280.0	180.0	234.15	298.81	354.96	12.07	1.16	−4.55
72	230.0	300.0	350.0	167.93	278.84	184.55	−4.15	1.19	−4.96
73	330.0	380.0	300.0	316.54	379.0	303.98	13.46	1.0	−3.98
74	230.0	320.0	350.0	233.77	318.89	352.47	−3.77	1.11	−2.47
75	320.0	300.0	370.0	323.8	299.11	373.46	−3.8	0.89	−3.46
76	240.0	160.0	160.0	246.62	159.04	160.55	−6.62	0.96	−0.55
77	310.0	180.0	260.0	294.1	179.03	264.22	15.9	0.97	−4.22
78	150.0	230.0	370.0	154.54	228.7	373.97	−4.54	1.3	−3.97
79	360.0	320.0	360.0	365.15	318.84	358.01	−5.15	1.16	1.99
80	250.0	180.0	240.0	255.25	179.01	240.11	−5.25	0.99	−0.11
81	310.0	280.0	240.0	294.62	278.83	243.68	15.38	1.17	−3.68
82	220.0	150.0	280.0	223.86	149.01	284.23	−3.86	0.99	−4.23
83	180.0	360.0	150.0	180.53	359.1	151.9	−0.53	0.89	−1.9

Continue on next page

Table B.5 – continued from previous page

	Ground truth			Estimated			Error		
#	Width	Height	Lenght	Width	Height	Lenght	Width	Height	Lenght
84	280.0	190.0	290.0	283.58	188.97	293.68	−3.58	1.03	−3.68
85	260.0	310.0	320.0	266.73	309.26	327.11	−6.73	0.75	−7.11
86	240.0	380.0	340.0	243.38	379.06	345.62	−3.38	0.94	−5.62
87	390.0	170.0	180.0	395.24	168.94	183.28	−5.24	1.06	−3.28
88	370.0	280.0	260.0	377.25	278.89	261.97	−7.25	1.11	−1.97
89	320.0	310.0	190.0	319.79	308.92	193.12	0.21	1.08	−3.12
90	250.0	310.0	160.0	255.31	308.82	165.47	−5.31	1.18	−5.47
91	180.0	380.0	240.0	187.85	379.33	248.3	−7.85	0.67	−8.3
92	270.0	330.0	330.0	274.93	328.98	331.25	−4.93	1.02	−1.25
93	220.0	190.0	150.0	222.79	188.9	150.18	−2.79	1.1	−0.18
94	330.0	150.0	350.0	327.14	149.02	357.63	2.86	0.98	−7.63
95	340.0	150.0	230.0	356.47	339.07	322.77	−3.04	0.95	−3.88
96	350.0	340.0	320.0	343.04	149.05	233.88	−6.47	0.93	−2.77
97	150.0	250.0	310.0	149.98	248.75	316.29	0.02	1.25	−6.29
98	270.0	230.0	280.0	268.55	228.87	284.46	1.45	1.13	−4.45
99	160.0	240.0	290.0	163.77	238.74	298.22	−3.77	1.26	−8.22
100	170.0	260.0	320.0	169.69	259.1	321.62	0.31	0.89	−1.62
101	200.0	320.0	370.0	203.58	318.89	372.94	−3.58	1.11	−2.94
102	370.0	260.0	320.0	374.53	258.98	322.42	−4.53	1.02	−2.42
103	320.0	390.0	270.0	319.58	389.0	269.95	0.42	1.0	0.05
104	330.0	330.0	180.0	327.75	329.15	184.71	2.25	0.85	−4.71
105	330.0	220.0	330.0	331.17	218.77	334.24	−1.17	1.23	−4.24
106	280.0	350.0	370.0	275.05	349.05	372.66	4.95	0.95	−2.66
107	270.0	320.0	160.0	263.66	349.18	167.27	−1.39	1.18	−2.34
108	260.0	350.0	160.0	271.39	318.82	162.34	−3.66	0.81	−7.27
109	150.0	270.0	250.0	156.15	269.39	254.14	−6.15	0.61	−4.13
110	360.0	190.0	200.0	363.14	188.74	203.74	−3.14	1.26	−3.74
111	260.0	160.0	250.0	261.02	159.35	254.41	−1.02	0.65	−4.41
112	270.0	150.0	180.0	280.43	148.93	181.81	−10.43	1.07	−1.81
113	250.0	170.0	350.0	255.24	169.11	354.2	−5.24	0.89	−4.2
114	180.0	180.0	270.0	183.29	179.12	272.58	−3.29	0.88	−2.58
115	360.0	390.0	150.0	333.63	388.95	153.22	26.37	1.05	−3.22
116	270.0	160.0	210.0	383.54	179.01	302.58	−2.6	0.7	−2.5
117	380.0	180.0	300.0	272.6	159.3	212.5	−3.54	0.99	−2.58
118	250.0	260.0	230.0	248.82	259.06	240.12	1.18	0.94	−10.12
119	350.0	250.0	380.0	353.91	248.94	382.84	−3.91	1.06	−2.84
120	280.0	150.0	150.0	284.03	149.41	158.21	−4.03	0.59	−8.21
121	250.0	170.0	260.0	251.15	168.77	261.56	−1.15	1.23	−1.56
122	210.0	260.0	290.0	214.0	259.04	292.75	−4.0	0.96	−2.75
123	330.0	150.0	270.0	333.93	149.16	275.76	−3.93	0.84	−5.76
124	150.0	240.0	390.0	155.0	238.93	395.02	−5.0	1.07	−5.02

Continue on next page

Table B.5 – continued from previous page

	Ground truth			Estimated			Error		
#	Width	Height	Lenght	Width	Height	Lenght	Width	Height	Lenght
125	230.0	210.0	150.0	236.54	209.1	153.43	−6.54	0.91	−3.43
126	370.0	190.0	210.0	373.99	189.11	214.05	−3.99	0.89	−4.05
127	290.0	380.0	320.0	290.07	378.97	326.39	−0.07	1.03	−6.39
128	340.0	380.0	170.0	342.28	378.99	174.29	−2.28	1.01	−4.29
129	260.0	240.0	180.0	263.07	239.05	184.78	−3.07	0.95	−4.78
130	350.0	180.0	280.0	354.97	178.71	285.09	−4.97	1.29	−5.09
131	320.0	200.0	330.0	325.91	198.83	333.81	−5.91	1.17	−3.81
132	220.0	170.0	380.0	223.89	169.15	384.52	−3.89	0.85	−4.52
133	360.0	190.0	300.0	364.54	189.21	303.65	−4.54	0.79	−3.65
134	350.0	220.0	370.0	355.24	219.18	373.92	−5.24	0.82	−3.92
135	290.0	190.0	280.0	293.93	189.0	281.59	−3.93	1.0	−1.59
136	320.0	330.0	340.0	317.79	329.19	338.3	2.21	0.81	1.7
137	360.0	170.0	250.0	365.81	168.89	253.75	−5.81	1.11	−3.75
138	330.0	200.0	320.0	333.46	198.68	320.46	−3.46	1.32	−0.46
139	180.0	200.0	150.0	185.22	199.2	150.73	−5.22	0.81	−0.73
140	240.0	330.0	270.0	239.36	329.09	274.35	0.64	0.91	−4.35
141	220.0	340.0	290.0	223.73	339.05	294.54	−3.73	0.95	−4.54
142	310.0	150.0	200.0	303.42	148.95	200.99	6.58	1.05	−0.99
143	270.0	370.0	330.0	272.48	369.06	332.07	−2.48	0.94	−2.07
144	380.0	390.0	230.0	383.76	389.1	232.34	−3.75	0.9	−2.34
145	380.0	250.0	370.0	384.9	249.26	373.17	−4.9	0.74	−3.17
146	260.0	180.0	350.0	269.27	179.04	353.57	−9.27	0.96	−3.57
147	300.0	190.0	190.0	302.42	188.91	191.94	−2.42	1.09	−1.94
148	190.0	290.0	310.0	193.98	288.72	313.29	−3.98	1.28	−3.29
149	360.0	230.0	190.0	204.61	238.9	331.41	−3.99	1.23	−3.05
150	200.0	240.0	330.0	363.99	228.77	193.05	−4.61	1.09	−1.41
151	360.0	370.0	210.0	354.96	368.89	214.13	5.05	1.11	−4.13
152	190.0	150.0	310.0	164.86	309.23	182.48	−3.41	0.83	−5.63
153	160.0	310.0	180.0	193.41	149.17	315.63	−4.87	0.77	−2.48
154	240.0	350.0	380.0	242.64	349.04	384.52	−2.64	0.96	−4.52
155	200.0	170.0	270.0	185.18	319.17	213.28	−4.02	1.03	−4.88
156	180.0	320.0	210.0	204.02	168.97	274.88	−5.18	0.83	−3.28
157	250.0	310.0	200.0	245.04	309.02	211.35	4.96	0.98	−11.35
158	220.0	230.0	310.0	225.16	228.8	312.95	−5.16	1.2	−2.95
159	160.0	300.0	180.0	163.93	298.75	181.99	−3.93	1.25	−1.99
160	260.0	390.0	240.0	255.06	389.05	244.7	4.94	0.95	−4.7
161	390.0	220.0	350.0	394.04	249.13	391.76	−5.09	1.17	−3.52
162	390.0	250.0	390.0	395.09	218.83	353.52	−4.04	0.87	−1.76
163	160.0	320.0	320.0	162.5	318.95	317.67	−2.5	1.05	2.33
164	240.0	260.0	280.0	243.84	259.0	281.59	−3.84	1.0	−1.59
165	380.0	220.0	310.0	385.06	219.09	316.43	−5.06	0.91	−6.43

Continue on next page

Table B.5 – continued from previous page

	Ground truth			Estimated			Error		
#	Width	Height	Lenght	Width	Height	Lenght	Width	Height	Lenght
166	350.0	160.0	300.0	152.17	319.14	309.29	11.79	0.73	−4.8
167	150.0	320.0	310.0	338.21	159.27	304.8	−2.17	0.86	0.71
168	270.0	160.0	190.0	274.11	158.99	194.13	−4.11	1.01	−4.13
169	180.0	170.0	390.0	182.15	168.84	396.07	−2.15	1.16	−6.07
170	360.0	220.0	390.0	362.86	218.64	394.36	−2.86	1.37	−4.36
171	330.0	150.0	280.0	334.34	148.78	283.74	−4.34	1.22	−3.75
172	220.0	230.0	250.0	227.08	228.9	245.77	−7.07	1.1	4.23
173	230.0	300.0	170.0	232.4	298.91	174.02	−2.4	1.09	−4.01
174	360.0	220.0	230.0	355.18	278.99	202.24	1.44	0.91	−3.28
175	350.0	280.0	200.0	358.56	219.09	233.28	−5.18	1.01	−2.24
176	280.0	200.0	290.0	282.24	199.27	292.36	−2.24	0.73	−2.36
177	380.0	150.0	370.0	363.37	338.96	283.2	−5.07	0.73	−4.53
178	350.0	190.0	370.0	385.07	149.27	374.53	−2.86	0.98	−4.1
179	360.0	340.0	280.0	352.86	189.02	374.1	−3.37	1.04	−3.2
180	150.0	340.0	380.0	145.29	338.74	395.31	4.71	1.26	−15.31
181	390.0	360.0	160.0	401.37	359.19	164.29	−11.37	0.81	−4.29
182	340.0	390.0	280.0	343.94	388.73	284.4	−3.94	1.27	−4.4
183	340.0	250.0	250.0	345.18	248.92	253.69	−5.18	1.07	−3.69
184	370.0	180.0	170.0	374.33	179.24	171.03	−4.33	0.76	−1.03
185	350.0	270.0	250.0	352.27	269.05	255.85	−2.27	0.95	−5.85
186	230.0	160.0	270.0	282.8	288.92	176.71	−3.14	0.8	−3.28
187	280.0	290.0	170.0	233.14	159.2	273.28	−2.8	1.08	−6.71
188	260.0	330.0	330.0	259.55	328.87	332.68	0.45	1.13	−2.68
189	380.0	330.0	310.0	362.35	378.74	394.19	−3.32	0.79	−3.53
190	360.0	380.0	390.0	383.32	329.21	313.53	−2.35	1.26	−4.19
191	210.0	190.0	390.0	214.18	189.17	390.03	−4.18	0.83	−0.03
192	180.0	170.0	200.0	183.23	168.84	198.22	−3.23	1.16	1.78
193	240.0	260.0	390.0	273.61	339.15	364.51	−6.01	0.84	−2.77
194	270.0	340.0	360.0	273.71	359.06	254.47	−3.61	0.85	−4.51
195	270.0	360.0	250.0	246.01	259.16	392.77	−3.71	0.94	−4.47
196	170.0	280.0	280.0	178.11	278.82	291.14	−8.11	1.18	−11.14
197	300.0	290.0	360.0	304.06	289.07	363.65	−4.06	0.93	−3.65
198	340.0	290.0	250.0	342.45	288.87	253.64	−2.45	1.13	−3.65
199	160.0	170.0	340.0	164.84	169.0	340.24	−4.84	1.0	−0.24
200	320.0	370.0	330.0	321.52	368.63	333.56	−1.52	1.37	−3.56
201	310.0	250.0	160.0	213.93	328.97	271.18	−1.85	0.88	−4.37
202	270.0	200.0	230.0	311.85	249.12	164.37	−2.69	0.96	−3.46
203	210.0	330.0	270.0	272.69	199.04	233.46	−3.93	1.03	−1.18
204	210.0	320.0	230.0	223.38	318.91	230.52	−13.38	1.09	−0.52
205	300.0	230.0	180.0	302.16	228.91	184.73	−2.16	1.09	−4.73
206	350.0	240.0	230.0	242.81	298.9	163.44	−1.05	0.84	−3.89

Continue on next page

Table B.5 – continued from previous page

	Ground truth			Estimated			Error		
#	Width	Height	Lenght	Width	Height	Lenght	Width	Height	Lenght
207	240.0	300.0	160.0	351.05	239.16	233.89	−2.81	1.1	−3.44
208	320.0	170.0	220.0	311.65	168.98	223.96	8.35	1.01	−3.96
209	360.0	210.0	350.0	361.34	208.89	353.79	−1.34	1.11	−3.79
210	270.0	280.0	260.0	273.6	278.9	263.96	−3.61	1.1	−3.96
211	190.0	150.0	390.0	194.56	148.82	394.16	−4.56	1.19	−4.16
212	360.0	220.0	360.0	364.34	218.9	359.52	−4.34	1.1	0.48
213	180.0	220.0	210.0	320.53	328.48	253.0	−6.54	1.23	−5.06
214	320.0	330.0	250.0	173.38	279.3	359.95	−0.53	1.52	−3.0
215	170.0	280.0	360.0	186.54	218.77	215.06	−3.38	0.7	0.05
216	180.0	260.0	270.0	181.56	258.6	263.08	−1.56	1.4	6.92
217	300.0	390.0	230.0	301.2	388.96	233.39	−1.2	1.04	−3.39
218	370.0	200.0	270.0	368.06	199.1	273.23	1.94	0.9	−3.23
219	220.0	180.0	210.0	223.49	179.19	216.72	−3.49	0.81	−6.72
220	270.0	300.0	350.0	273.2	298.77	362.7	−3.2	1.23	−12.7
221	350.0	290.0	390.0	353.31	289.12	394.71	−3.31	0.88	−4.71
222	300.0	340.0	310.0	302.51	338.91	313.21	−2.51	1.09	−3.21
223	170.0	170.0	200.0	172.49	168.78	202.84	−2.49	1.22	−2.84
224	350.0	160.0	180.0	354.66	158.58	186.7	−4.67	1.42	−6.7
225	220.0	190.0	220.0	292.6	379.02	314.2	−2.42	0.95	−0.96
226	290.0	380.0	310.0	255.48	228.89	382.24	−2.6	0.98	−4.2
227	250.0	230.0	380.0	222.42	189.05	220.96	−5.49	1.11	−2.24
228	300.0	290.0	170.0	303.21	289.12	178.2	−3.21	0.88	−8.21
229	310.0	370.0	180.0	316.79	369.24	183.46	−6.79	0.76	−3.46
230	170.0	320.0	230.0	172.51	319.01	233.56	−2.51	0.99	−3.56
231	330.0	220.0	320.0	333.78	218.98	324.74	−3.78	1.02	−4.74
232	380.0	220.0	340.0	382.33	219.12	341.67	−2.33	0.88	−1.67
233	310.0	210.0	180.0	183.18	338.98	186.09	4.35	1.03	−4.88
234	210.0	190.0	220.0	305.65	208.97	184.88	0.33	0.86	−3.76
235	180.0	340.0	180.0	209.67	189.14	223.76	−3.18	1.02	−6.09

Table B.5: Results size estimation on simulation-data, all dimensions in mm.

Appendix C

Results Robotic bin packing

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
1	2	0	0
2	3	0	2
3	2	0	0
4	3	0	0
5	2	0	0
6	3	0	1
7	1	1	0
8	2	0	0
9	2	0	0
10	2	0	1
11	2	0	2
12	1	0	1
13	4	0	0
14	2	0	0
15	2	0	1
16	4	0	0
17	4	1	0
18	3	0	1
19	1	0	0
20	3	0	2
21	1	0	0
22	1	0	2
23	2	0	0
24	3	0	0
25	6	0	0

Continue on next page

Table C.1 – continued from previous page

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
26	2	0	0
27	3	0	1
28	2	0	1
29	3	0	3
30	4	0	0
31	2	0	1
32	3	0	0
33	1	0	0
34	3	0	1
35	4	2	0
36	3	0	3
37	1	0	1
38	2	0	0
39	2	0	0
40	2	0	2
41	4	0	0
42	2	0	1
43	1	0	2
44	2	0	1
45	3	0	0
46	1	0	2
47	2	0	1
48	4	0	2
49	0	0	2
50	0	0	1
51	1	0	1
52	3	0	1
53	2	1	2
54	2	0	2
55	1	0	0
56	4	0	0
57	6	0	1
58	2	0	0
59	3	0	0
60	6	0	1
61	1	0	1
62	1	0	0
63	2	0	1
64	3	0	3
65	3	0	0
66	1	2	0
67	3	0	0

Continue on next page

Table C.1 – continued from previous page

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
68	1	0	1
69	2	0	2
70	5	0	0
71	4	0	1
72	2	0	0
73	2	0	1
74	4	0	0
75	1	0	2
76	3	0	0
77	1	0	0
78	4	0	1
79	3	1	0
80	1	0	0
81	2	0	1
82	2	0	2
83	2	0	0
84	1	0	2
85	1	0	2
86	2	0	2
87	4	0	0
88	1	0	1
89	2	0	0
90	2	0	0
91	2	0	0
92	3	0	1
93	2	0	0
94	2	0	2
95	1	0	1
96	3	0	0
97	1	0	0
98	4	0	0
99	2	4	0
100	1	0	3
101	3	0	1
102	1	0	0
103	1	0	3
104	2	0	1
105	1	0	2
106	4	1	0
107	1	0	1
108	2	0	1
109	2	0	1

Continue on next page

Table C.1 – continued from previous page

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
110	3	0	1
111	4	0	1
112	1	0	2
113	3	0	2
114	3	1	0
115	1	0	0
116	2	1	1
117	3	0	0
118	2	0	0
119	3	0	0
120	3	1	0
121	1	0	2
122	2	0	0
123	1	0	1
124	2	0	0
125	2	0	1
126	1	0	1
127	0	0	0
128	2	0	0
129	6	0	1
130	1	0	0
131	2	0	0
132	2	0	0
133	2	0	0
134	2	0	0
135	1	0	2
136	2	0	0
137	1	0	1
138	1	0	0
139	2	0	2
140	2	0	0
141	3	0	1
142	2	0	0
143	2	0	0
144	2	0	0
145	4	0	1
146	2	0	0
147	3	0	1
148	0	0	0
149	1	0	0
150	2	1	0
151	2	0	0

Continue on next page

Table C.1 – continued from previous page

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
152	2	0	1
153	2	0	1
154	2	0	2
155	3	0	0
156	1	0	1
157	2	0	0
158	2	0	0
159	3	0	1
160	1	0	0
161	4	0	0
162	4	0	1
163	2	0	0
164	2	1	2
165	2	0	0
166	3	0	0
167	2	0	2
168	2	0	3
169	1	0	0
170	5	0	0
171	3	0	0
172	4	0	1
173	3	0	1
174	3	0	2
175	2	0	0
176	1	0	1
177	1	0	0
178	1	0	0
179	2	0	1
180	3	1	1
181	3	0	2
182	1	0	1
183	1	0	0
184	2	0	0
185	2	0	2
186	2	0	0
187	6	0	0
188	3	0	2
189	2	0	1
190	3	0	0
191	2	0	1
192	3	0	1
193	3	0	2

Continue on next page

Table C.1 – continued from previous page

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
194	1	0	2
195	2	0	1
196	2	0	0
197	2	0	1
198	2	1	0
199	3	1	3
200	1	0	0

Table C.1: Number of unpacked boxes for the dataset with maximum height of 600 mm.

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
1	8	0	1
2	2	0	0
3	2	0	2
4	1	0	1
5	2	0	1
6	3	0	1
7	1	0	1
8	3	0	2
9	5	0	3
10	2	0	0
11	3	0	0
12	2	0	1
13	3	0	0
14	4	0	1
15	1	0	0
16	3	0	1
17	4	0	6
18	4	0	6
19	1	0	0
20	3	0	2
21	1	0	0
22	1	0	1
23	3	0	1
24	2	0	3
25	3	0	0
26	4	0	0
27	5	0	0
28	6	0	1
29	4	0	1
30	6	0	0
31	7	0	3
32	3	1	0
33	3	0	2
34	6	0	1
35	2	0	0
36	5	0	0
37	1	0	0
38	4	1	0
39	3	0	1
40	3	0	3
41	4	0	2
42	2	0	0
43	0	0	0

Continue on next page

Table C.2 – continued from previous page

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
44	3	0	3
45	2	0	2
46	4	0	1
47	1	0	3
48	5	0	0
49	4	0	1
50	5	0	2
51	3	0	0
52	2	0	2
53	2	0	0
54	4	0	0
55	3	0	2
56	2	0	3
57	6	0	2
58	2	0	0
59	3	0	0
60	4	0	1
61	3	0	1
62	5	0	1
63	4	1	2
64	1	0	3
65	5	0	2
66	4	0	3
67	3	0	0
68	3	1	3
69	3	0	2
70	3	0	1
71	1	0	1
72	2	0	1
73	2	0	0
74	2	0	2
75	3	0	0
76	3	0	0
77	3	0	3
78	5	0	1
79	4	0	3
80	3	0	1
81	4	0	0
82	3	0	3
83	2	0	3
84	1	0	2
85	3	0	1

Continue on next page

Table C.2 – continued from previous page

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
86	1	0	2
87	4	0	1
88	3	0	3
89	1	0	3
90	3	1	1
91	2	0	1
92	2	0	3
93	2	0	0
94	1	0	1
95	4	0	1
96	4	0	0
97	2	2	1
98	4	0	0
99	4	0	0
100	1	0	1
101	5	0	1
102	3	0	1
103	2	0	3
104	2	2	3
105	6	0	0
106	4	0	6
107	6	0	2
108	4	0	1
109	1	0	0
110	1	0	1
111	0	0	3
112	1	0	0
113	3	0	1
114	5	0	0
115	4	0	3
116	3	0	0
117	2	0	0
118	3	0	1
119	4	0	1
120	6	0	0
121	1	0	0
122	3	0	0
123	2	0	3
124	4	0	3
125	4	0	1
126	3	0	3
127	3	0	0

Continue on next page

Table C.2 – continued from previous page

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
128	1	0	2
129	4	0	0
130	5	0	2
131	2	0	2
132	5	0	2
133	2	0	1
134	3	1	0
135	2	0	1
136	4	1	1
137	3	0	1
138	3	0	3
139	2	0	0
140	6	1	0
141	3	0	0
142	4	1	0
143	1	0	1
144	3	0	1
145	4	1	0
146	0	0	3
147	5	0	1
148	3	0	1
149	3	0	1
150	3	2	0
151	2	0	0
152	4	0	1
153	4	0	1
154	4	0	0
155	5	0	1
156	3	0	1
157	3	0	1
158	2	0	1
159	4	0	1
160	3	0	1
161	9	0	0
162	3	0	0
163	2	0	1
164	3	1	1
165	6	0	2
166	1	0	0
167	4	0	1
168	6	0	1
169	5	0	1

Continue on next page

Table C.2 – continued from previous page

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
170	1	0	1
171	1	0	1
172	3	0	1
173	5	1	1
174	5	3	0
175	4	0	1
176	1	1	0
177	3	0	1
178	3	0	0
179	2	0	1
180	4	0	1
181	5	0	1
182	2	0	3
183	2	0	1
184	2	0	1
185	4	0	1
186	4	2	0
187	4	0	1
188	1	1	0
189	1	0	1
190	2	1	0
191	4	0	1
192	4	0	1
193	2	0	1
194	2	0	1
195	2	0	1
196	3	0	1
197	5	1	0
198	6	3	0
199	3	0	1
200	2	1	1

Table C.2: Number of unpacked boxes for the dataset with maximum height of 800 mm.

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
1	0	0	3
2	3	0	3
3	3	0	2
4	6	3	1
5	1	0	1
6	11	0	0
7	5	0	3
8	4	1	2
9	6	0	1
10	6	0	2
11	6	0	0
12	7	0	3
13	4	0	3
14	4	0	2
15	3	0	1
16	3	0	2
17	11	0	0
18	7	0	7
19	1	0	2
20	3	0	1
21	3	0	1
22	5	0	3
23	3	0	0
24	3	0	1
25	6	0	1
26	3	1	1
27	5	0	1
28	5	0	2
29	7	0	3
30	7	0	9
31	5	0	5
32	11	0	0
33	5	0	0
34	6	1	1
35	10	0	6
36	5	0	3
37	5	0	1
38	2	0	2
39	3	0	3
40	6	0	0
41	2	0	0
42	9	0	3
43	9	0	9

Continue on next page

Table C.3 – continued from previous page

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
44	6	0	6
45	8	0	6
46	5	0	5
47	5	1	1
48	3	0	2
49	5	0	2
50	2	0	0
51	2	0	2
52	8	0	3
53	6	0	1
54	6	0	1
55	4	0	2
56	5	0	1
57	7	0	6
58	2	0	3
59	3	0	3
60	12	0	1
61	7	1	7
62	4	0	2
63	8	0	1
64	4	0	1
65	6	0	3
66	5	0	0
67	5	0	3
68	5	0	2
69	5	0	5
70	1	0	2
71	7	0	3
72	1	0	1
73	3	0	0
74	9	0	12
75	9	0	2
76	6	0	0
77	9	0	0
78	5	0	0
79	4	0	1
80	7	0	0
81	6	0	2
82	1	0	3
83	4	0	3
84	6	0	1
85	5	0	1

Continue on next page

Table C.3 – continued from previous page

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
86	5	0	1
87	4	0	1
88	11	0	2
89	3	1	1
90	5	0	2
91	5	0	0
92	7	0	3
93	7	0	9
94	5	0	0
95	11	0	0
96	5	0	3
97	6	1	1
98	10	0	6
99	5	0	3
100	5	0	1
101	2	1	1
102	3	0	2
103	6	0	0
104	2	0	2
105	9	0	0
106	9	0	9
107	6	0	6
108	8	0	8
109	5	0	5
110	5	2	2
111	3	0	2
112	5	0	2
113	2	0	0
114	2	0	2
115	8	0	8
116	6	0	0
117	6	0	1
118	4	0	2
119	5	0	3
120	7	0	2
121	2	0	3
122	3	0	3
123	12	0	0
124	7	0	1
125	4	0	1
126	8	0	1
127	4	0	1

Continue on next page

Table C.3 – continued from previous page

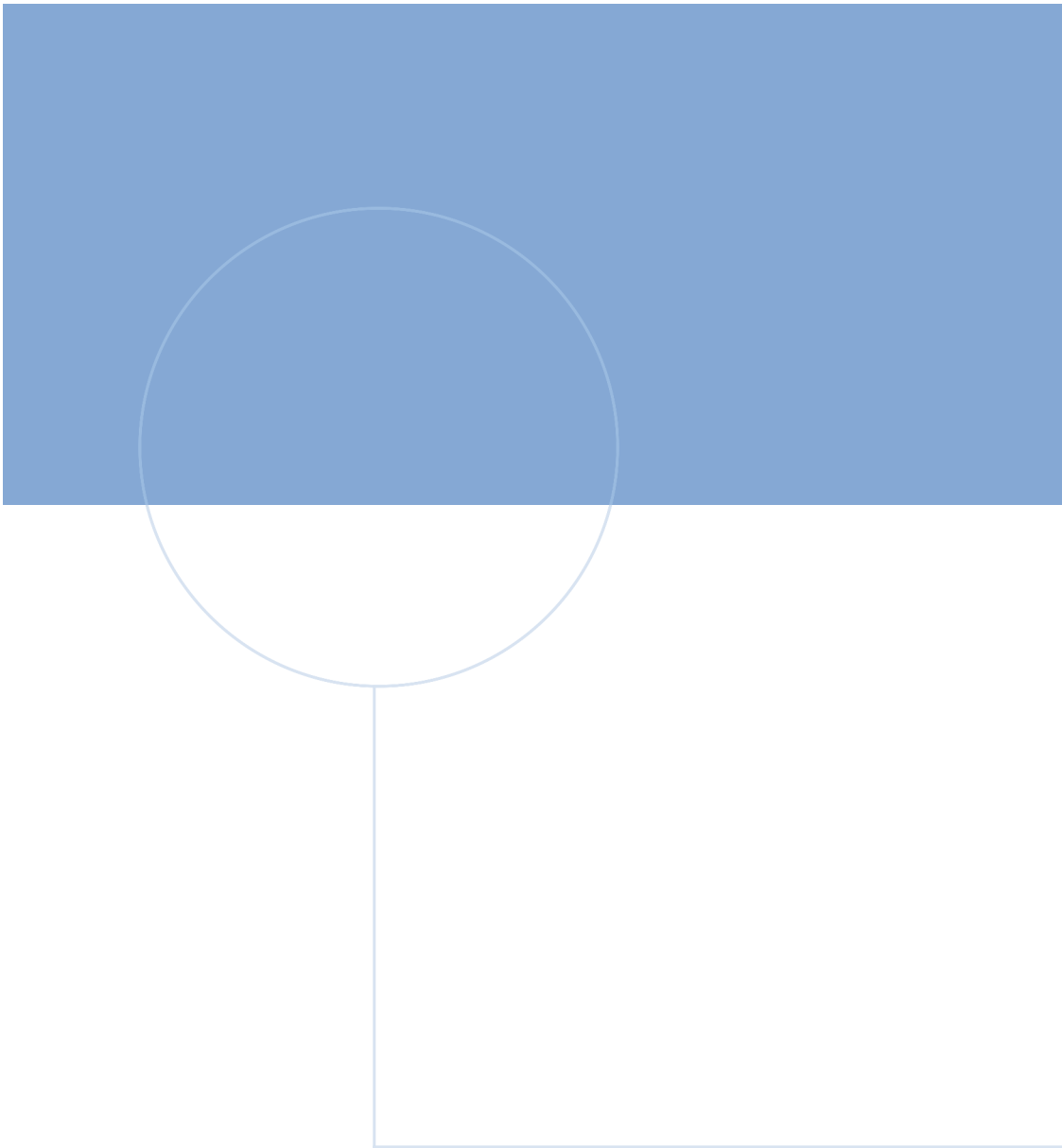
	Heuristic packing strategy	Bin packing problem	Robotic bin packing
128	6	0	3
129	5	0	0
130	5	0	2
131	5	0	2
132	5	0	5
133	1	0	2
134	7	0	3
135	1	1	1
136	3	0	2
137	9	0	9
138	9	0	2
139	6	0	6
140	9	0	0
141	5	0	2
142	4	0	3
143	7	0	2
144	6	0	2
145	1	0	3
146	4	0	3
147	6	0	1
148	5	0	1
149	5	0	1
150	4	0	3
151	11	0	2
152	3	0	1
153	5	0	1
154	5	0	1
155	7	0	3
156	7	0	9
157	5	0	5
158	11	0	0
159	5	0	0
160	6	1	1
161	10	0	10
162	5	0	3
163	5	0	1
164	2	1	1
165	3	0	1
166	6	0	2
167	2	0	0
168	9	0	1
169	9	0	1

Continue on next page

Table C.3 – continued from previous page

	Heuristic packing strategy	Bin packing problem	Robotic bin packing
170	6	0	6
171	8	0	6
172	5	0	5
173	5	3	3
174	3	0	2
175	5	0	2
176	2	0	0
177	2	0	1
178	8	0	3
179	6	0	0
180	6	0	1
181	4	0	2
182	5	0	0
183	7	0	7
184	2	0	2
185	3	0	3
186	12	0	1
187	7	1	2
188	4	0	0
189	8	0	1
190	4	0	1
191	6	0	3
192	5	0	1
193	5	0	2
194	5	0	1
195	5	0	5
196	1	0	2
197	7	0	3
198	1	1	1
199	3	0	0
200	9	0	12

Table C.3: Number of unpacked boxes for the dataset with maximum height of 1200 mm.



NTNU
Norwegian University of
Science and Technology