University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Information Technology - Automation and Signal Processing	Spring semester, 2020 Open
Author: Sigurd Ytterstad	
Programme coordinator: Karl Skretting Supervisor(s): Karl Skretting and Torfi Thorhallsson	
Title of master's thesis: Picking Products from Distribution Containers by Object Detection and Occlusion Estimation	
Credits: 30	
Keywords: Computer Vision, Point Clouds, Object Detection, Object Picking, Robot, Occlusion Estimation	Number of pages: 61 + supplemental material/other: 28 + attached file (code.7z) Stavanger, 28/07/2020 date/year

University of Stavanger

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Picking Products from Distribution Containers by Object Detection and Occlusion Estimation

Master's Thesis in Information Technology - Automation and Signal Processing

by

Sigurd Ytterstad

Internal Supervisor

Karl Skretting

External Supervisor

Torfi Thorhallsson

June 28, 2020

Abstract

There is a significant increase in e-commerce, and groceries are entering the online platform. With this development, and with customers wanting to change their shopping habits from brick-and-mortar stores to the online platform, automation is needed to relieve the manual labor required for picking products and making order fulfillment effective.

The objective of this thesis is to create a solution for picking products out of distribution containers and examine the challenges and limitations of the proposed solution. The system installation used in the approach is a robot with five degrees of freedom, three for navigating the X, Y, Z-coordinates, and two for rotating the end-effector.

The proposed solution is developed with some assumptions. Two of these being that the products are not stacked in height and that the cardboard is not covering the top of the product. The approach is to capture a depth image of the scene and to apply a scale-invariant feature transform to detect and create a bounding box of the product. The region contained by the bounding box is compared to a reference image, and the color differences of the images are used for cardboard estimation. With the estimated cardboard, combined with the depth information from the camera, a collision map is created for collision detection.

Two experiments are conducted. In the first experiment, the products are reset to the initial states for each pick, and a path planner based on rapidly exploring random tree is used to create the robot's path for retrieving the product. The second experiment is based on the same approach, but the product is skewed and picked with Cartesian control.

From the results, and within the assumptions and constraints of this thesis, a Cartesian control is sufficient for retrieving the products, and the cardboard estimation proves robust for a delimited range of products. However, future analyses are needed to determine the range of products the solution applies to, and it is suggested to do some research into deep neural networks, to see if it can outperform the proposed solution.

Video recording of an excerpt of the products used in the experiments is available from [1, Expiration date: August 1, 2021].

Acknowledgements

This thesis concludes my Master's degree in Automation and Signal Processing from the University of Stavanger.

I want to start off by thanking my supervisors Karl Skretting and Torfi Thorhallsson for giving me valuable guidance throughout this thesis.

I also want to thank Pickr.ai for the opportunity for this project, with a special thanks to Mikal Berge for providing technical assistance when working on their installation, as well as advice whenever needed.

Lastly, I would like to send my gratitude to Simon Lennart Austnes and Harald Thirud Skutvik for proof-reading my thesis.

Contents

Abbreviations

APC	**A**mazon **P**icking **C**hallenge
BIT	**B**ath **I**nformed **T**rees
DOF	**D**egree **O**f **F**reedom
FOV	**F**ield **O**f **V**iew
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
HSV	**H**ue **S**aturation **V**alue
IoU	**I**ntersection **o**ver **U**nion
$\mathbf{IoU_{CD}}$	**I**ntersection **o**ver **U**nion for **C**ardboard **D**etection.
$\mathbf{IoU_{PD}}$	**I**ntersection **o**ver **U**nion for **P**roduct **D**etection.
JSON	**J**ava**S**cript **O**bject **N**otation
OMPL	**O**pen **M**otion **P**lanning **L**ibrary
PCL	**P**oint **C**loud **L**ibrary
RGB-D	**R**ed **G**reen **B**lue - **D**epth
R-CNN	**R**egion - **C**onvolutional **N**eural **N**etworks
RANSAC	**RAN**dom **SA**mple **C**onsensus
ROI	**R**egion **O**f **I**nterest
RL	**R**einforcement **L**earning
RRT	**R**apidly Exploring **R**andom **T**ree
SIFT	**S**cale-**I**nvariant **F**eature **T**ransform
SR	**S**uccess **R**ate
TN	**T**rue **N**egative
TP	**T**rue **P**ositive

Chapter 1

Introduction

1.1 About Pickr.ai

Pickr.ai [2] proposed this master's thesis in cooperation with the University of Stavanger. Pickr.ai is a start-up company founded in 2016, located in Stavanger, Norway.

Pickr.ai is an automation company specializing in order fulfillment. The company is still in the start-up phase, thereby the goal and direction of development might adjust or change depending on the need and demands of the customers.

One of Pickr's objectives is to make their product cost-efficient and to enable companies to transfer to an online market with an automated solution and make a profit on products with a low margin for profit.

1.2 Motivation

In the world, as well as Norway, e-commerce is increasing rapidly. From 2016 to 2019, e-commerce had a growth of 35.2 % [1] in Norway [4][2].

Not only is e-commerce growing, but a new target is entering the online platform, namely the grocery industry. According to Swisslog [5], in the United States, the e-groceries have not been affected in the same way as the retail stores when looking at e-commerce growth, with $\approx 3\%$ of the groceries being shopped online in 2019.

[1] The data was retrieved from Statistics Norway. It includes retail sales over the internet and mail. More information about the content is found in [3], with code: 47.91.

[2] There is a known bug with the link. It sometimes directs to an empty table. The fix, if encountering this problem, is to click the link two times (or copy-paste into the browser twice).

However, from a report by Brick Meets Click from 2018 [6], online grocery spending has increased by 22% from 2017 to 2018. Furthermore, in an analysis by Fabric from 2019 [7], looking at the grocery industry in the United States, the increasing popularity of online groceries is a fact. Not only are the customers willing to pay more to get same-day delivery on their products, but they are also willing to switch to retailers that offer this feature.
For the stores to offer these online features, there is a need for automation as the company is losing money on the manually picked products, and these can be reduced by, e.g., introducing an automated micro-fulfillment system [7].

Pickr.ai is interested in looking deeper into the warehouse picking of groceries, and the possibilities of picking the products directly from the distribution package. The task of picking out of distribution packages is a task many warehouses do manually, and the general picking-problem is of interest for research, e.g., Zhu et al. [8] and Shao et al. [9].

1.3 System Overview

The system used in this thesis is a small-scale test system used for testing new functionality and creating proof-of-concept demos without risk of damaging the stable release of Pickr's system. Figure 1.1 shows the system used.

The robot is a gantry robot [3] with 3-DOF (degrees of freedom) operating the X-Y-Z axis and a 2-DOF robotic arm. The robotic arm operating in the system workspace is displayed in figure 1.3. Along with the joint connecting the end effector, an RGB-D (Red, Green, Blue, and Depth) camera is mounted. The camera is positioned to capture the whole shelf compartment, as shown in figure 1.2.

Robot Constraints

The robot has 5-DOF, three prismatic joints, and two rotational joints. However, the joints have limitations in extensions and rotations, leading to restrictions in the reachable workspace.

The extension in the Z-direction is limited to 50cm, resulting in a reach of $\approx$ 20cm within the shelf's depth. For the rotational joints, the rotational freedom is illustrated in figure

[3] Gantry robot (also referred to as Cartesian or linear robots) is a robot working in an X-Y-Z coordinate system (three prismatic joints).

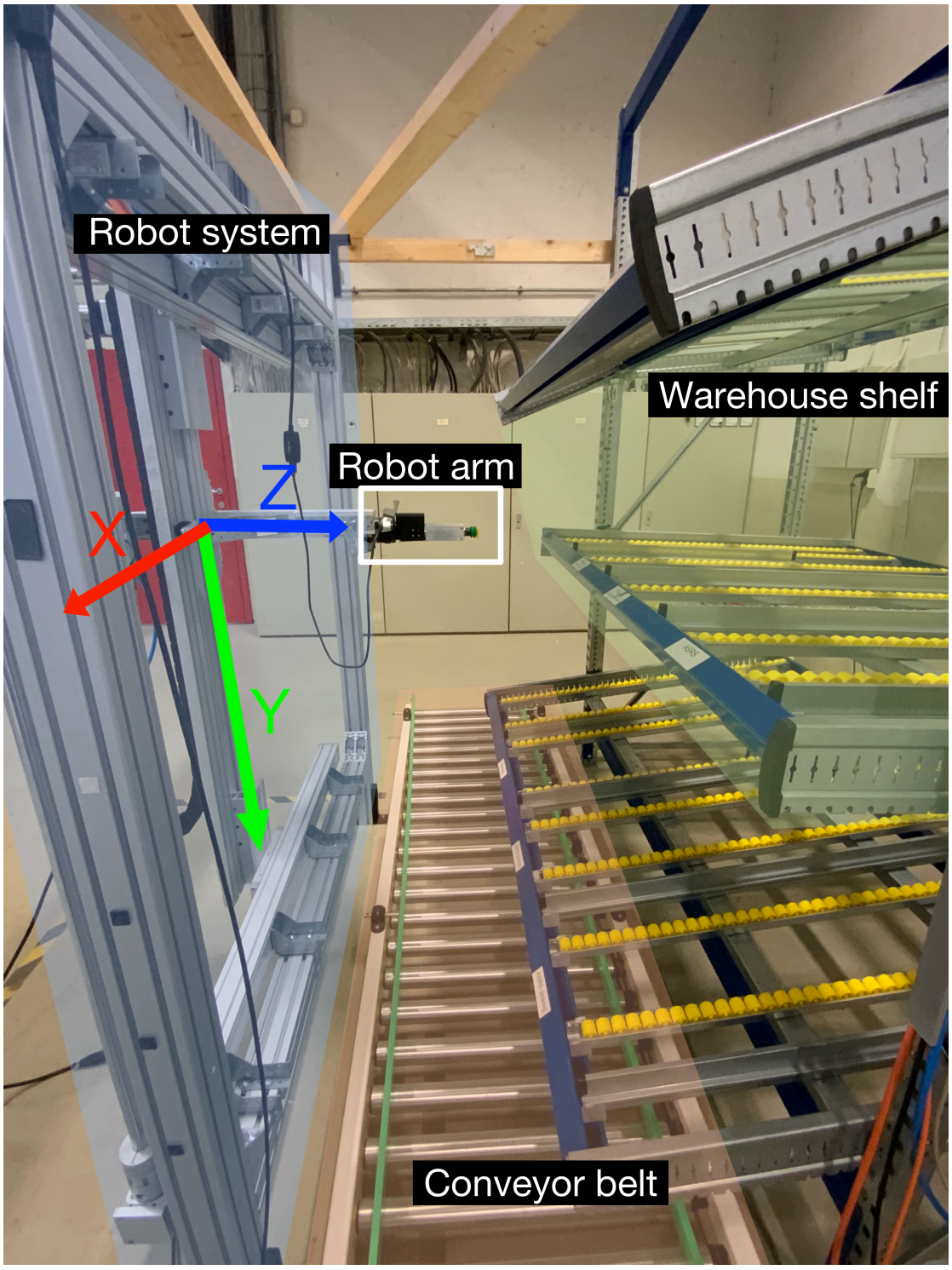
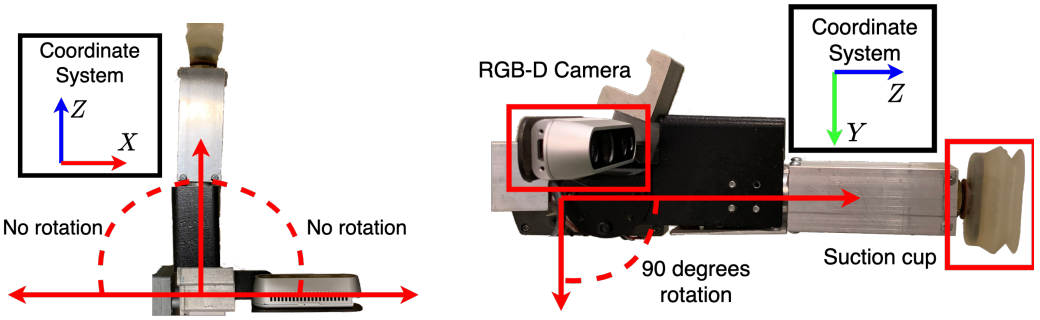


Figure 1.1: The figure displays the system in which this thesis revolves around. On the left is the robot operating the system (highlighted in blue), and on the right is the warehouse shelf (highlighted in green). On the bottom is a conveyor belt (highlighted in red), but is not used in this thesis.

1.3 and 1.4, and gives 90° rotation vertically, 0° rotation horizontally and ±90° around the Z-axis (around itself).

Figure 1.2: The figure displays an image captured by the RGB-D camera. The dotted red line marks the boundary of one of the shelf compartments. The figure only displays the RGB part of the image.



(a) Top view of the robotic arm.

(b) Side view of the robotic arm.

Figure 1.3: The figure illustrates the rotational freedom of the robotic arm. Figure (a) shows the horizontal rotation, and figure (b) the vertical rotation. The coordinate system is also illustrated, where the Z-axis is marked in blue, X-axis in red, and Y-axis in green.

End Effector

The end effectors used in this setup were suction cups. The choice of using suction cups was made by Pickr, but they are considered to be one of the most versatile and robust end effectors to pick objects [10]. As shown in figure 1.4, the end effector has a rotation of 180° around the Z-axis.

Two different suction cups were used. The default suction cup is shown in figure 1.5(a), it is oval, with the majority of the area in the vertical direction. Having a large area in the vertical direction allows for more stable level picks because the gravity is pushing the object down. In those cases, the additional stability of a larger suction cup is beneficial. However, there was experimented with skewed products. With those products, it proved beneficial to change the suction cup to a kind with some flexibility and contraction of

the bellow, as this allowed for additional pushing to achieve a sufficient vacuum. This suction cup is seen in figure 1.5(b).

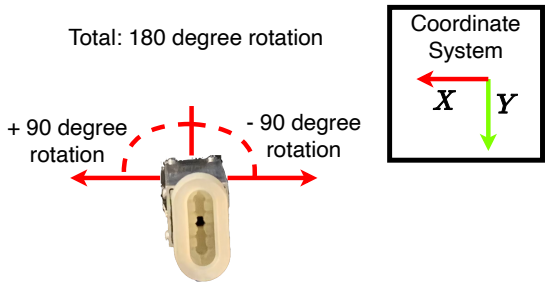
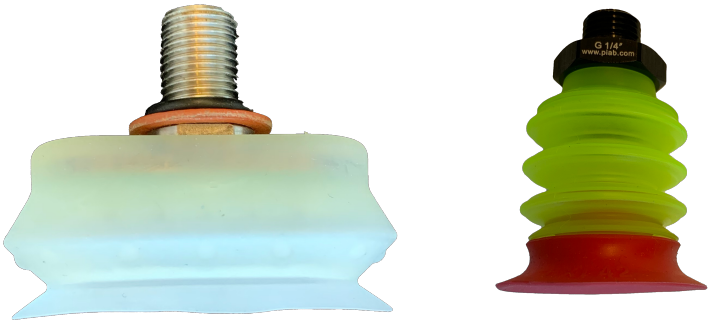


Figure 1.4: The figure displays the orientation of the suction cup with the coordinate system. The end effector has a rotation of ±90°.



(a) The default suction cup.

(b) Suction cup for picking skewed products..

Figure 1.5: The figure displays the two different suction cups used in this setup. The suction cup in figure (a) was most suitable for products positioned perpendicular to the end-effector. From (b) the suction cup for picking skewed products is shown, this suction cup has a larger bellow with more flexibility and the possibility for contraction, making it more suitable for skewed products. The drawback is more sensitivity to the weight of the products.

Constraints of the Reachable Workspace

Because the end-effector has only 90° rotation in the vertical direction (from level to top-down), the end effector is not capable of reaching above 0°. Because of this, the shelf has a negative slope of ≈ 6°, compensating for the limitation, and avoiding potential ambiguity if the shelf was level-oriented[4].

Since the end-effector have no rotation in the horizontal direction (around the Y-axis), all pick-points with an orientation around this axis is unreachable. However, because

[4]When the shelf has an 0° angle, the product could have a minor orientation, e.g., 0.1°, resulting in an unreachable orientation.

of flexibility in the suction cup's bellow, and allowing some pushing to get a suitable suction region, picking products with a rotation in the horizontal direction is possible. The implementation of the solution is described in section 3.5.1.

1.4 Problem Definition

The objective of this thesis is to look into a solution of a robotic system for picking grocery products out of distribution packages, illustrated by figure 1.6.

Figure 1.6: The figure displays grocery products placed on the shelf within the original distribution package. The objective of this thesis is to create a proof-of-concept system for picking products out of the distribution packages.

Picking out of distribution packages is a desired feature as it reduces the time and manual labor spent picking from shelves before the automatic process begins.

However, picking products out of shelves introduces several challenges compared to picking products in a workspace without obstructions. It is necessary to look at the robot path planning for retrieving the products, as the robot path is not necessarily a straight line. Furthermore, a collision map is necessary when generating the robot path because of potential collisions with the cardboard, the shelf, or other products.

Constraints of the Problem

Some assumptions and limitations were introduced to narrow the scope of the problem.

- The products are not stacked on top of each other: If stacked products are considered, it expands the problem to evaluate whether or not the product for retrieval is at the top. If the product is at the bottom, the products above might fall when the product is removed.

- Cylindrical products are not evaluated: Cylindrical products are not ideal with the proposed object detection method. Pickr.ai already has a suitable object detection method for cylindrical objects.
- Products need to have a solid surface: Plastic containers, e.g., bags of chips, have not been evaluated. For those products, the shape can change and is unreliable in the proposed cardboard detection method.
- Small products can not be picked: Small products, with not large enough area for the suction cup to access, have not been evaluated.
- The cardboard can not cover the top of the product: This constraint was introduced since the camera is pointing directly at the shelf, and detecting the cardboard's small width was not optimal with the proposed solution.

Main Challenges

The problem can be divided into three main parts.

- Detecting the products and estimate the cardboard: The first step in this approach was to find a method to distinguish between products and cardboard. The product for extraction had to be detected, and the cardboard had to be estimated.
- Generate and implement an accurate collision map: For the robot to extract the product without collision, obstructions had to be estimated and used in a collision map. The cardboard, shelf, and the other products are all objects that the robot should avoid.
- Generate a working robot-path for retrieving the product: When the steps mentioned above are complete, a robot-path had to be made.

The result of this thesis should evaluate the following issues:

- How accurate and reliable is the cardboard and product detection?
- What is the success rate when picking products?
- If failure, are there any obvious reasons?
- How is the performance of the solution when picking skewed products?
- Is Cartesian control sufficient for picking products in distribution packages?
- What can be improved upon the proposed solution?

1.5 Thesis Outline

Chapter 2 - Background:

This chapter introduces relevant information about the current state of the problem that this thesis revolves, in addition to related work and research by others, and the current approach by Pickr.

Chapter 3 - Solution Approach:

This chapter covers the approach used to solve the problem. The chapter explains the implementation and uses the methods described in Chapter 2.

Chapter 4 - Experimental Evaluation:

This chapter covers the setup of the conducted experiments, with the corresponding results. The evaluation metrics used in the evaluation are also described.

Chapter 5 - Discussion and Future Directions:

This chapter covers the interpretations of the results of the conducted experiments, with suggested future work and possible other directions.

Chapter 6 - Conclusion:

This chapter describes the conclusion of this thesis.

Chapter 2

Background

This chapter introduces the background to the work in section 3: Solution Approach. The background includes related work in similar domains, the baseline of Pickr's system, and methods relevant for the solution approach.

2.1 Current Approach/Baseline

The current baseline for picking products in Pickr's system does not involve picking from distribution containers, but instead, picking out of standardized boxes, illustrated by figure 2.1.

Figure 2.1: The figure displays the products placed in standardized boxes. The robotic system is the same as described in section 1.3: System Overview.

The background to the current system is based, among others, on the work by Eriksen [11] master's thesis in 2017.

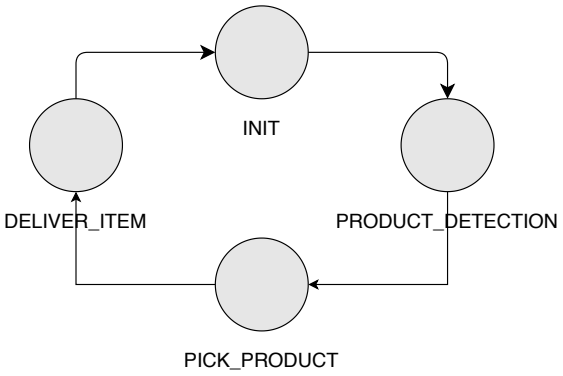


Figure 2.2: Simplified layout of the current system.

Figure 2.2 shows a simplified flow chart of the system. The camera mounted on the robot captures an RGB-D image of the container with the products, illustrated in 2.1. The current system is mainly focused on cylindrical shaped objects and detects a suitable surface for the suction cup. Because there is no occlusion to account for, the path is a straight line.

2.2 Related Work

The problem related to robotic pick-and-place is a highly popular research area, and on system level, the problem includes robot-vision, object recognition, and robotic path planning [12]. Evaluation of such systems has been done in the Amazon Picking Challenge (APC) and the competition held at the International Conference on Intelligent Robots and Systems (IROS). Those competitions, although the APC has discontinued, aims to provide a benchmark for the variety of problems involved in a robotic pick-and-place system [13].

In the APC, the goal was to use state-of-the-art technology for warehouse automation [14]. The winner of the 2017 APC challenge, Zeng et al. [15], used an RGB-D camera to capture an image of the scene. From the image, different picking options were evaluated, and the option satisfying their criteria was chosen. When the object was retrieved, it was positioned in front of the camera, before identifying the object by matching it against numerous product images.

Another popular method to solve the picking problem is through Reinforcement Learning (RL). It was used in the paper by Breyer et al. [16], Zeng et al. [17], and was initially

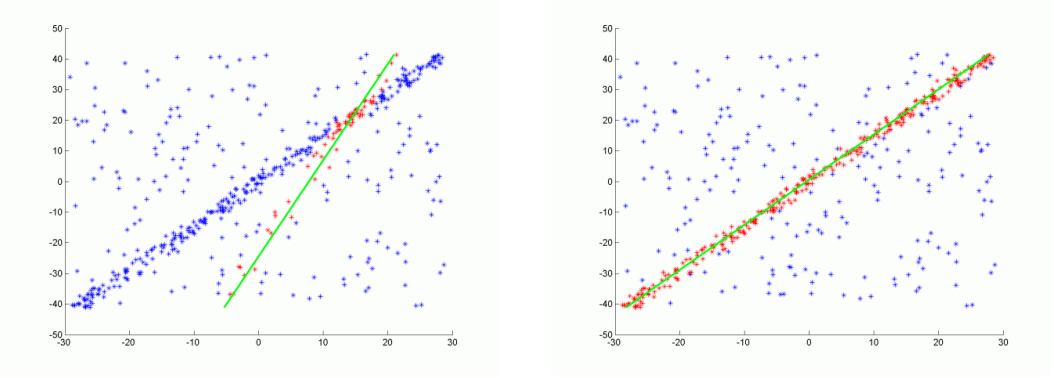
suggested as an interesting approach to solving the problem in this thesis. RL is part of the Machine Learning category and is used in various applications, ranging from playing video games to complex robotic maneuvering [18]. RL is a "trial and error" based approach that increase or decrease a score depending on the behavior and ought to gain the highest possible score [19, p. 331]. More information about RL is found in [19, chap. 17].

Within object detection, Faster R-CNN, Ren et al. [20], is considered the state-of-the-art [21]. A method that extends the Faster R-CNN is the Mask R-CNN, He et al. [22], which in addition to detect objects, creates a segmented mask around the object. Another popular method used in object detection is YOLO (You Only Look Once), and was used in the object detection problem by Wang et al. [23], as it proved to have some improved speed compared to Faster R-CNN.

There are numerous path planners to choose from when it comes to generating a path from point A to point B. Sampling-based motion planning is a popular path planning method within robotics [24], and one alternative within the sampling scheme is Rapidly Exploring Random Trees (RRT), introduced by LaValle and Kuffner [25], LaValle [26]. Several other implementations based upon RRTs are, e.g., Bath Informed Trees (BIT*), Gammell et al. [27], and RRT-Connect, Kuffner and LaValle [28]. More information about motion planning, BIT*, and RRT-Connect is in section 2.8.

2.3 Random Sample Consensus

Random Sample Consensus (RANSAC), Fischler and Bolles [29], is an iterative algorithm for model fitting. Given some data and a model (e.g., a line, plane, or cylinder), RANSAC ought to find a model that maximizes the number of inliers (points residing within some distance from the model). The algorithm starts by randomly choosing a set of points sufficient for creating an instance of the model (e.g., two points for a line), then counting the number of points residing within some distance threshold from the line. This process is repeated X times, and the instance of the model with the highest number of inliers is given as the final estimate, Szeliski [30, p. 281-282] and Hartley and Zisserman [31, p. 117]. The RANSAC algorithm is illustrated in figure 2.3.



(a) (b)

Figure 2.3: The figure shows two iterations of the RANSAC algorithm for estimating a line through some data. Figure (a) show one iteration, and (b) shows another iteration with the best fit. The red dots are the points from the data considered inliers. `The figure is reprinted in unaltered form from Wikimedia commons, File:RANSAC LINIE Animiert.gif, released into the public domain.`

2.4 Color Space: Hue, Saturation and Value

The Hue, Saturation, and Value (HSV) color space is another way to represent colors than the traditional Red, Green, and Blue (RGB) color space. The difference lies in the separation of the components. In HSV, the Hue represents the colors, the Saturation describes the intensity or purity of the color, and the Value represents the brightness [32, p. 1300]. This is illustrated in figure 2.4.

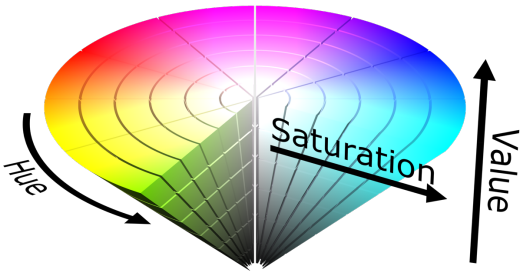


Figure 2.4: The figure shows illustration of the HSV color space. `The figure is reprinted in unaltered form from Wikimedia commons, File:HSV color solid cone.png, licensed under CC BY-SA 3.0.`

The benefit of the HSV color space in the application in this thesis is that it separates the Hue, Saturation, and Value, and where the color information is contained in the Hue component. Because the purpose is to evaluate color differences between two images, this separation enables for a more convenient comparison of the images compared to

the RGB color space, which has three components, all containing color information (RGB).

However, if the Saturation value approaches 0, there is little to no information about the color (Hue) in the image, giving the image only different shades of gray. If evaluating images based on Hue, this introduces a problem when the Saturation is low. When the Saturation is below some threshold, it is possible to evaluate the pixel value based on the Value (brightness) [33].
One way to decide this threshold is based on equation 2.1, where V is the Value (brightness), and th_{sat} is the suggested threshold for the Saturation, as suggested by Sural et al. [33]. The equation is rewritten to suit the value range used in this thesis, having H, S, and V $\in [0 - 255]$.

$$\text{th}_{\text{sat}}(\text{V}) = 255 - 0.8\text{V} \tag{2.1}$$

The threshold th_{sat} gives information about whether the information in the pixel is more distinct by the Hue or the Value, giving the following evaluation:

$$\text{Evaluate} = \begin{cases} \text{Hue}, & \text{if S} \geq \text{th}_{\text{sat}}(\text{V}) \\ \text{Value(brightness)}, & \text{otherwise} \end{cases}$$

2.5 Binary Image Morphology

Binary morphology in image processing are operations that change the shape of binary images [30, p. 112]. In this thesis, the operation was used to remove structural outliers in the cardboard estimate.
Morphological operations use a structuring element to change the shape of an image. The structuring element has a specific shape (e.g., rectangle or ellipse), and has a defined origin, which acts as a reference point for the structuring element (it points to the pixel to be changed). When the structuring element traverses through the binary image, the region contained by the structuring element is evaluated, and depending on the values and the morphological operation, the pixel of the origin is changed [34, p. 75-77].

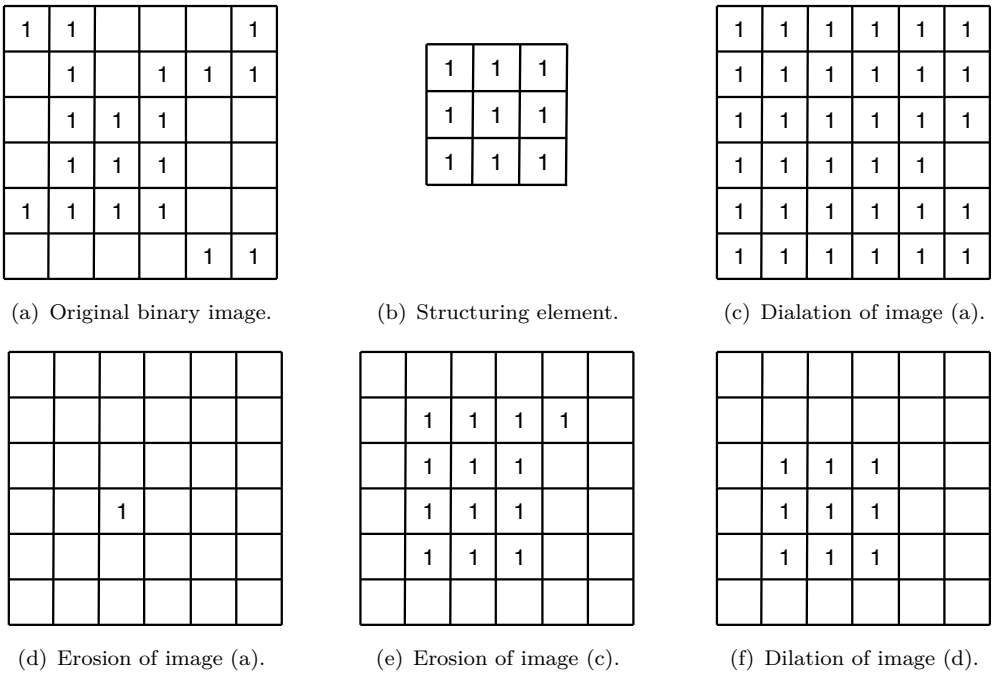
2.5.1 Dilation and Erosion

There are two basic morphological operations, *dilation* and *erosion*, and they can be applied together to obtain other operations [35, sec. 7.2.].
Dilation can be thought of as an OR operation. If any positive value of the structuring element overlaps with a positive value of the binary image, the value of the binary image is changed to a positive value. Otherwise, it is set to zero [34, p. 78].

Erosion, on the other hand, can be thought of as an AND operation. All of the structuring element's positive values have to overlap with the binary image's positive values in order for the evaluated pixel to be set positive. Otherwise, it is set to zero [34, p. 79].

2.5.2 Opening and Closing

Combining *erosion* and *dilatation*, using the same structuring element, creates the new operations: *opening* and *closing*. *Opening* is the use of *erosion* followed by *dilation*, while *closing* is the use of *dilation* followed by *erosion* [35, sec. 7.3.].
The desired outcome by using these operations, in this thesis, is to fill holes in the binary image and smooth the outer edges. Figure 2.5 displays the mentioned four operations.



(a) Original binary image. (b) Structuring element. (c) Dialation of image (a).

(d) Erosion of image (a). (e) Erosion of image (c). (f) Dilation of image (d).

Figure 2.5: The figure displays the use of *Dilation*, *Erosion*, *Opening*, and *Closing*, of an image (a) with structuring element (b). The operation shown in (e) is the same as *closing* (a), and the operation in (f) is achieved by *opening* (a). `The figure is inspired by Shapiro and Stockman [34, figure 3.13, p. 80]`.

2.6 Point Clouds

Point cloud, in general, is used to represent multi-dimensional points. In this implementation, the point clouds were used to represent points in three dimensions from captured images.

Regular images have (x, y) coordinates for each pixel, with color values representing them. Point clouds have (x, y, z) coordinates, with the z-value representing the depth (from the captured depth image).

With a 3D representation of the environment, it is possible to do image processing in traditional 2D, and use the processed data to make changes to the point cloud.

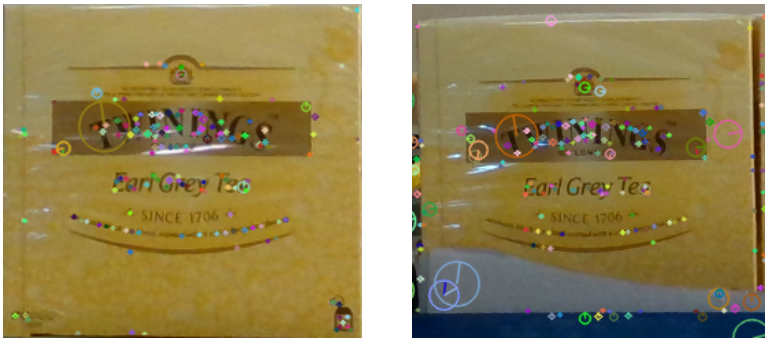
2.7 Scale-Invariant Feature Transform

Scale-invariant feature transform (SIFT) is an algorithm introduced by Lowe [36, 37].

The purpose of using this method is to identify the products in the scene. The identification is achieved by finding matching features in a reference image and corresponding matches in the captured scene image. These images will be referred to as "reference image" and "scene image" correspondingly.

2.7.1 Keypoints and Descriptors

The SIFT algorithm is used to find features, also called keypoints. Keypoints are as the name suggests points of interest in the image (e.g., a corner). These keypoints holds information about the location, scale, and orientation of the feature [37, p. 14], as shown in figure 2.6.



(a) Keypoints from reference image. (b) Keypoints from scene image.

Figure 2.6: The figure shows keypoints from the SIFT algorithm, with size and orientation (the circles in the images). The scene image (b) is cropped to only display one product.

To each of these keypoints a corresponding descriptor are computed. These descriptors are invariant to rotation, translation and scaling in the image domain [38, p. 1], and are used to compare the features.

2.7.2 Matching Images

The product detection is achieved by finding keypoints in the two images and using their descriptors to compare and match the images [38, p. 6]. The method used in this thesis to compare the keypoint descriptor was Fast Library for Approximate Nearest Neighbors (FLANN). This approach to estimate nearest neighbors is described in the article by Muja and Lowe [39].
Because some of the matches might not correspond to the product of interest, Lowe [37, p. 20] proposed to use a ratio test to reject false matches. The matching and detection of the product "tea" are seen in figure 2.7, with a bounding box placed around the product.

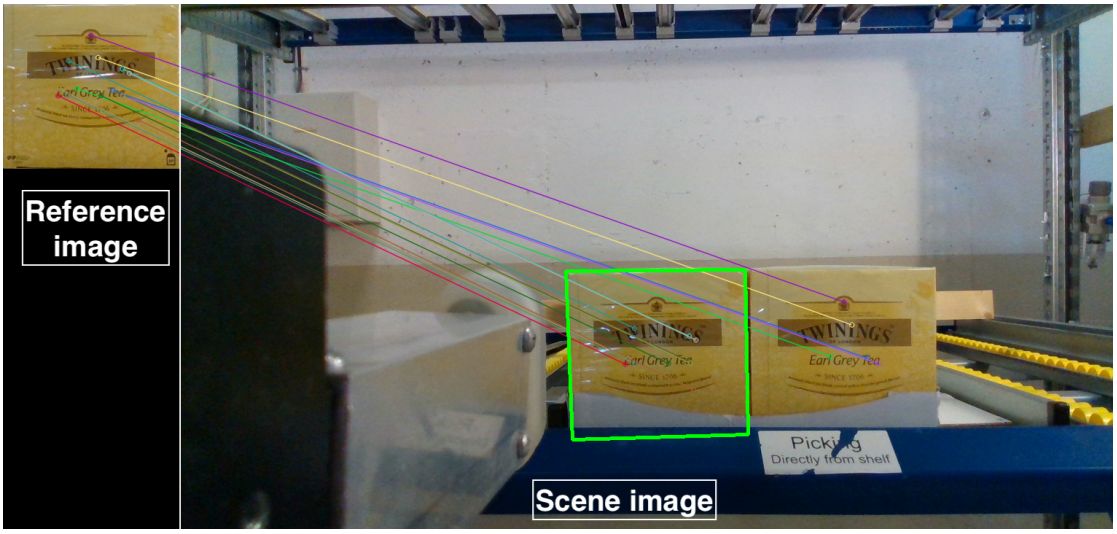


Figure 2.7: The figure illustrates the SIFT feature matching. The features in the reference image (left) are matched with those in the scene image (right). If there is a match, a bounding box is drawn around the product (bold green line in the right image).

2.7.3 Homography

Homography maps points between two images, represented by a 3×3 matrix with eight degrees of freedom, and requires four corresponding points from the images to be determined [31, p. 88]. The homography is used to calculate the bounding box from figure 2.7. In the same manner, the homography is used to remove projective distortion from the scene image, relative to the reference image, as will be described in section 3.3.2.

2.8 Motion Planning

This section describes motion planning, the planners used in this thesis, and an introduction to optimal motion planning. The configuration space Q is the possible configurations of the robot and consists of Q_{free}, configuration without obstacles, and Q_{obs}, configuration with obstacles [24, p. 2]. Figure 2.8 displays a sampled configuration space with a generated path. Where q_{init} is the start position, and q_{goal} is the target position.

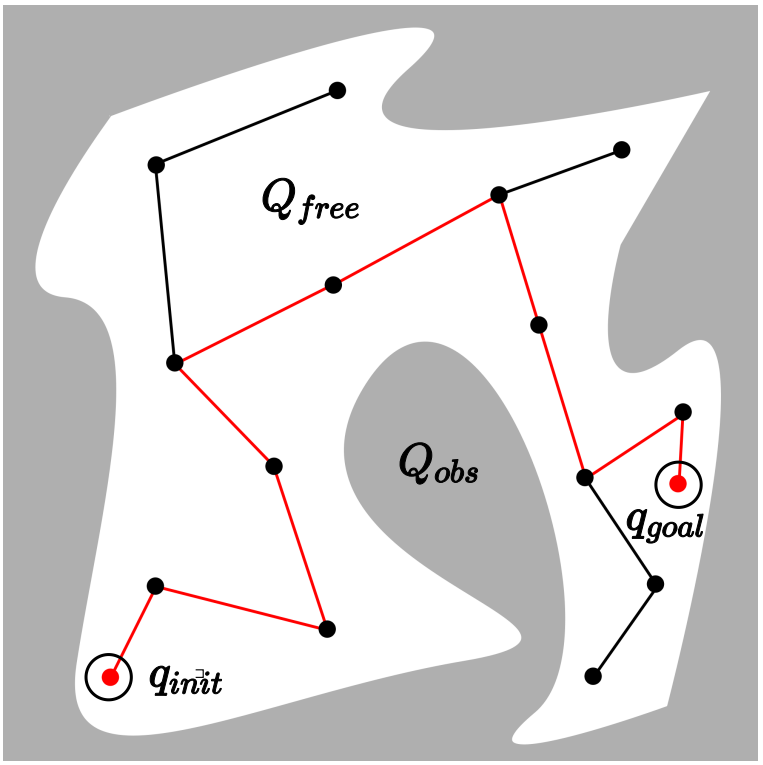


Figure 2.8: The figure illustrates a sampling approach for finding a path from q_{init} to q_{goal}. The path is marked with a red line. The black dots are the samples within Q_{free}, and the grey area is occupied space, Q_{obs}. The figure is only meant for illustration purposes and not meant for describing any specific path planning algorithm. The figure is reprinted and annotated with custom text from Wikimedia commons, File:Motion planning configuration space road map path.svg, licensed under CC BY-SA 3.0.

2.8.1 Sampling-Based Planners

The idea of sampling-based planners is to explore the Q-space with a sampling approach. The following two terms will be used, [24, p. 2-3]:

- *Probabilistic Completeness*: If a solution exists, the planner will find a solution when given infinite samples.

- *Asymptotic optimality*: The planner converges to the optimal solution when given infinite samples.

The terms describe some of the properties of the planners in this section. For more information about the subject Sampling-Based Planners, see [40, chap. 5], and Bekris and Shome [24].

Rapidly Exploring Random Tree (RRT)

Rapidly Exploring Random Tree (RRT) is a sampling-based motion planner introduced by LaValle and Kuffner [25], LaValle [26].
The RRT algorithm is probabilistic complete. However, it does not necessarily find the optimal path between two points. I.e., if a solution is found, continuing to explore Q_{free} does not alter the outcome, even if there is a shorter path among the samples.
Pseudocode of the RRT algorithm is shown in algorithm 1, inspired from Karaman and Frazzoli [41, p. 13], where q_{init} is the start state, q_{goal} is the target state, ϵ is the length for which the new node extends from the nearest node, and K is the number of iterations. The `Extend` function links the random point q_{random} with the neighboring node q_{near}, and finds a point extended ϵ along that path, creating the new node q_{new}, [28, p. 2].

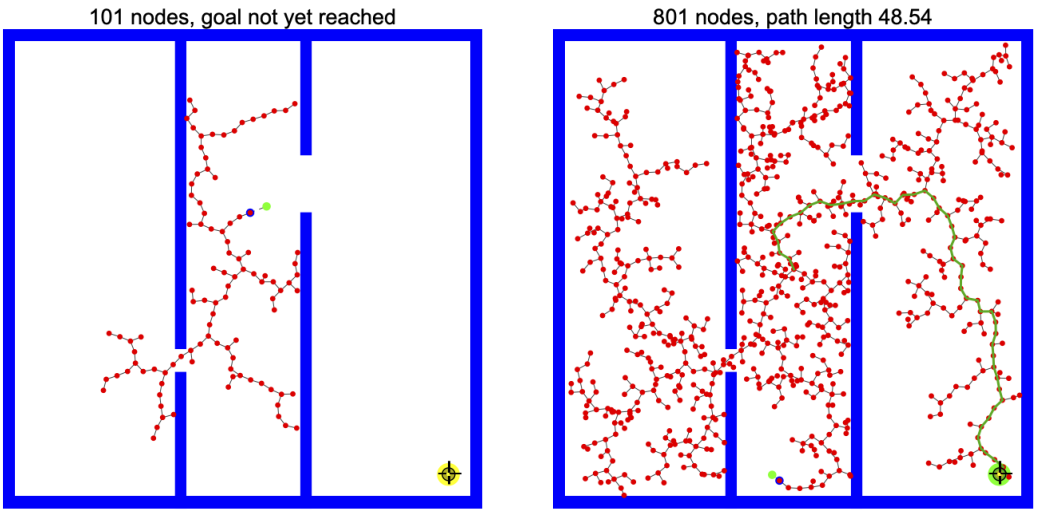
Algorithm 1 RRT($q_{init}, q_{goal}, \epsilon, K$)

1: $G(N, E)$ ▷ Initialize empty graph (G) containing Nodes (N) and Edges (E).
2: $G() \leftarrow q_{start}, q_{goal}$ ▷ Insert goal and start node to the graph.
3: **for** *iteration* : $1 \rightarrow K$ **do**
4: $q_{random} \leftarrow$ RandomPosition( ) ▷ Find a random position $q_{random} \in Q_{free}$.
5: $q_{near} \leftarrow G$.GetNearestNode(q_{random}) ▷ Get the node closest to q_{random}.
6: $q_{new} \leftarrow$ Extend($q_{random}, q_{near}, \epsilon$) ▷ Extend the new node with distance ϵ.
7: G.AddNode(q_{new}) ▷ Add the new node to the graph.
8: G.AddEdge(q_{near}, q_{new}) ▷ Add the edge between the nodes.
9: **end for**

Figure 2.9 shows the RRT algorithm after 100 and 800 iterations.

RRT-Connect

The path planning algorithm used in this thesis was the RRT-connect, Kuffner and LaValle [28]. The basic idea is that it builds two RRTs, one at the start position, q_{start} and one at the goal position, q_{goal}, and use a `Connect` heuristic. The `Connect` heuristic is an alternative to the `Extend` function used in the RRT algorithm 1. The `Connect` function iterates the `Extend` function until q_{random} is reached or until detecting an obstacle, [28, p. 3].



(a) RRT algorithm after 100 iterations (b) RRT algorithm after 800 iterations

Figure 2.9: The figure shows the RRT algorithm after 100 iterations (a) and 800 iterations (b). The different nodes are marked in red circles, and the goal is the yellow marker at the bottom right (a) and marked as green in (b). One new node is added for each iteration, starting with one node. In figure (b), a path has been found, marked by a green line. `The figure is generated from [42], licensed under: CC BY-NC-SA.`

2.8.2 Optimal Motion Planning

Optimal motion planning try to optimize some path. The optimization objective could be, e.g., generating the shortest path. In contrast, the non-optimal planner finds any feasible solution [40, p. 357]. One of such optimal planner is the RRT*, Karaman and Frazzoli [41]. The RRT* is an asymptotically optimal version of the RRT algorithm, and it is probabilistic complete [41, p. 20-21]. More information about optimal motion planning are available from [40, chap. 7.7].

Batch Informed Trees (BIT*)

Batch Informed Trees (BIT*), Gammell et al. [27], is a sampling-based optimal planner. The planner is probabilistic complete and asymptotically optimal. Simplified, the algorithm creates a graph [1] of Q_{free} and explores the graph (in the direction of q_{goal}) by a heuristic search [2]. When a solution is found, the batch is complete. The process is repeated with a denser graph and continued until a satisfied solution is reached [27, p. 3].

[1] The graph is a Random Geometric Graphs (RGG). It will not be explained here, but more information is found in [43].

[2] Heuristic search is, as quoted from [44]: *"a search strategy that attempts to optimize a problem by iteratively improving the solution based on a given heuristic function or a cost measure."*

A descriptive video illustrating the algorithm is available from [45], and more information about BIT* is found in Gammell et al. [27, 46].

Chapter 3

Solution Approach

This section introduces the tools and resources used in the solution approach, along with the implementation.

3.1 Tools and Resources

This section describes the different libraries and tools used in the implementation, in addition to the evaluation metric used to evaluate the experiment.

3.1.1 Labeling Tool: Labelbox

Labelbox [47] is a tool for labeling data. It was used to label all the images after the experiment for evaluating the performance of the product detection and cardboard estimation against some ground truth. The labeling process is described in section 4.1.2.

3.1.2 Robot Operating System

The Robot Operating System (ROS) [48] is a framework for developing robot software. It is not an operating system. Instead, it is a collection of tools and libraries to ease the troubles of working on different robotic platforms. ROS has a peer-to-peer structure, meaning that the system has several independent processes communicating through a master. It also supports several different programming languages so that the code can be written in, e.g., Python or C++ [49, 50].

A simple example is illustrated in figure 3.1, where the node `pub_node` is publishing some message. The messages are published on the topic `\some_topic`, and the other node `sub_node` is listening on the same topic and receives the information published.

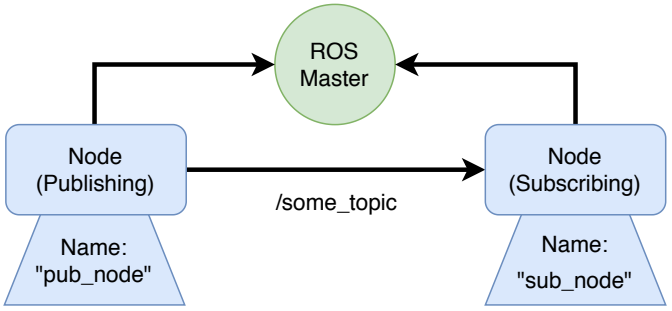


Figure 3.1: The figure illustrates the basic ROS setup. The ROS master manages and sets up communication between the nodes. The communication is in this illustration done through the topic `\some_topic`.

ROS Master

The ROS master is at the core of the system and manages all topics, nodes, and sets up communication [51].

ROS Node

ROS nodes are program modules that execute some operation. The nodes are independent (standalone program) and can be programmed in one programming language and still communicate with other nodes programmed in a different language [52].

ROS Topic

Topics are channels where information is transferred. A node publishes information on a topic (e.g.,`\some_topic` in figure 3.1), and a receiving node subscribes to the same topic [53].

3.1.3 Depth Camera

The camera used in the implementation is the Intel RealSense D435 [54]. The camera was already part of the implementation by Pickr, and have proved to be of satisfaction. The camera is a stereo vision depth camera capable of capture images in stereo depth and RGB [55, p. 11].

The camera has a Z-accuracy [1] $\leq 2\%$, within 2 meters, and 80 % of FOV [55, table. 4-9, p. 61]. I.e., within the workspace of this setup (ranging from 0 to 1 meter), the expected error is on the range of $0 - 20$mm.

3.1.4 Programming Languages

Pickr's existing system is mainly developed in C++, and some of the libraries used in this thesis have limited to no support for other languages. Therefore it was natural to use C++ for the implementation and development of code in this thesis.
However, for the experiment evaluation, Python was used to manage and evaluate the different images produced, including retrieving and sorting the images stored in Labelbox's cloud.

3.1.5 Product Information

Information about the products was received from the customer and saved in a JSON[2] file for quick retrieval. The JSON management tool used was RapidJSON [58]. The exception to the data retrieved from the customer is the reference image, which is captured manually and is described in section 3.3.1. In table 3.1, the information about the product "Tea" is displayed. This file will be referred to as *Product File*.

Table 3.1: The table illustrates the information available in the JSON file for each product when picking the items. "D-Pack" is short for "Distribution Package."

Type Of Information	Value	Data Type	Unit
Product Width	0.15	Double	Meter
Product Height	0.14	Double	Meter
Product Depth	0.078	Double	Meter
D-Pack Width	0.305	Double	Meter
D-Pack Height	0.144	Double	Meter
D-Pack Depth	0.402	Double	Meter
Reference Image	"/path/to/file.png"	String	N/A

3.1.6 Point Cloud Library

The Point Cloud Library (PCL) [59] is used for 2D/3D image and point cloud processing. In this thesis, it is mainly used for processing point clouds and doing 3D calculations.

[1] Z-accuracy, or absolute error, is the difference between the measured depth and the actual depth (ground truth) [56, p. 7].
[2] JSON is a lanugage-independent format for data exchange [57, chap. 2].

Furthermore, as stated by Rusu and Cousins [60]: *"PCL is a comprehensive free, BSD licensed, library for n-D Point Clouds and 3D geometry processing."* ROS also supports the library.

3.1.7 Open Source Computer Vision Library

Open Source Computer Vision Library (OpenCV) is *"an open source computer vision and machine learning software library"* [61], and has been used in this implementation for image processing in 2D. The OpenCV library is licensed under the 3-clause BSD License [62].

3.1.8 Motion Planning Framework: MoveIt!

MoveIt! [63] is a motion planning framework that runs on top of ROS, and are licensed by BSD License v3. MoveIt! provides motion, path planning, and collision checking used in this thesis.

3.1.9 Open Motion Planning Library

The Open Motion Planning Library (OMPL) [64], is a library consisting of state-of-the-art sampling-based motion planning algorithms, and was used with MoveIt! to generate robot-paths. The OMPL library is licensed under the BSD License [65].

3.1.10 Octrees and OctoMap

Octrees are a tree structure where each node, called voxel, can have eight children. Voxels are similar to pixels, only represented in 3D, and are mostly used to represent a three-dimensional space or volume. For using voxels to represent occupancy, the voxel could contain some binary information, e.g., whether or not the voxel represents occupied space. In octrees, the voxel is sub-divided into eight new voxels, as illustrated in figure 3.2, until some user limit is reached (the resolution of the tree). By the hierarchical structure of the octrees, if all children of a node have the same state, it can be represented by the parent node. Representing the tree by a node closer to the root reduces the number of nodes in the tree, thus reducing capacity and resources [66, p. 4].

The occupancy estimation method used in this thesis is OctoMap [67]. OctoMap is, as the title of the paper states: a *"Probabilistic 3D Mapping Framework Based on Octrees,"* developed by Hornung et al. [66].

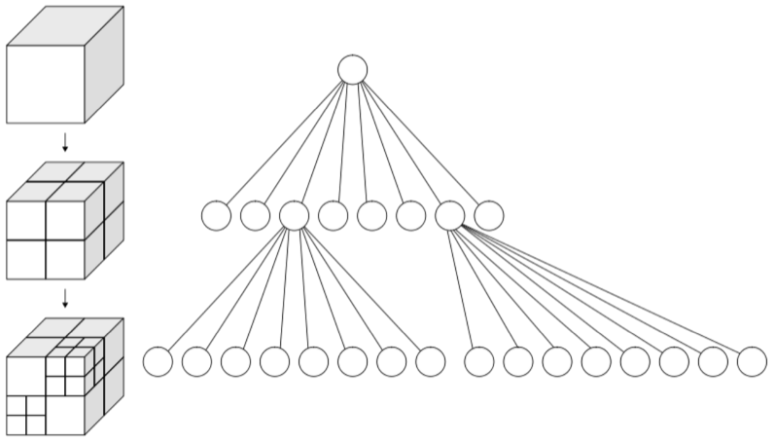


Figure 3.2: The figure illustrates the octrees subdivisions. Each node (circle in the image) has eight children nodes. `The figure is reprinted in unaltered from Wikimedia commons, File:Octree2.svg, licensed under CC BY-SA 3.0.`

OctoMap was used to generate a collision map in MoveIt!, and supports the use of point clouds as inputs. The library is available as a self-contained source distribution.

3.1.11 Visualization Tool: RViz

Rviz [68] is a visualization tool for ROS. It allows for a visual representation of the robotic world and simulation. It supports both ROS and MoveIt!.

3.2 Solution Overview

The developed program runs a state machine created by a "switch-case" in the main.cpp file. This state machine also describes the system in a simplified manner, as the essential functions are located within each state.
Figure 3.3 illustrates the state-machine of the developed program. The illustration contains the important modules in the program:

- Product Detection and Cardboard Estimation
- Collision Map Generation
- Robot Path Planning

These three parts are what is described in the rest of this chapter: "Solution Approach." The exception is the state `INIT`, which is the initialization state. Before transitioning from this state, the *Product File* is loaded, objects are instantiated, and ROS are initialized.

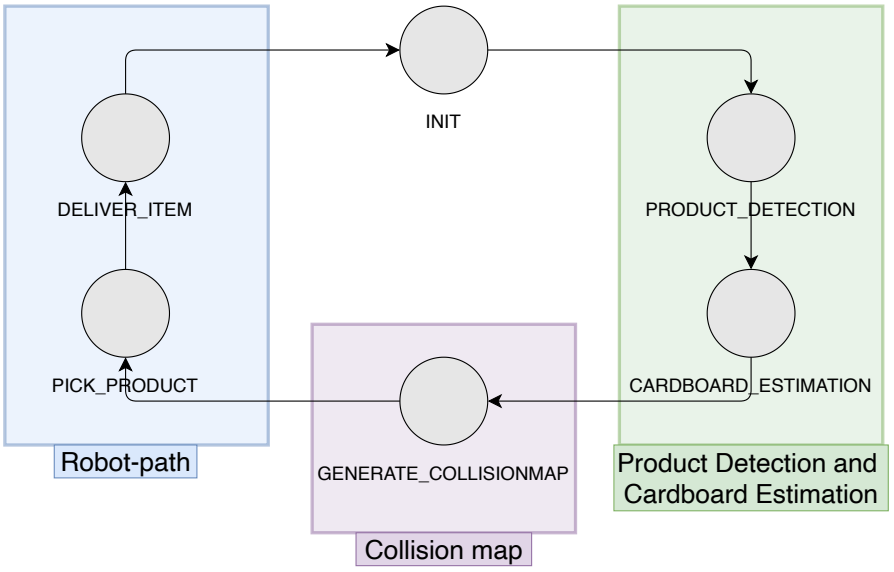


Figure 3.3: The figure displays an illustration of the state-machine. The first state (`INIT`) moves the robot to a predefined location (in this case, it is the position in front of the shelf) and loads information from the *Product File*. The next states are divided into the three categories described in this section.

3.3 Proposed Solution for Detecting the Product and Estimating the Cardboard

In this section, the proposed solution for detecting the product and estimating the cardboard is described. The purpose was first to detect the product and create a bounding box of the product. Then, estimate the cardboard covering the product by comparing the difference between the bounding box and a reference image. A simplified flow chart of the process described in this section is available in figure C.5. The cardboard estimation was used for collision detection, which is described in section 3.4.

3.3.1 Reference Image

The method used for product detection (SIFT) requires a reference image of the product in order to detect the product in the scene image. For product detection and cardboard estimation to deal with fewer dissimilarities, the reference image is captured under the same environment and with the same camera. The reason for this, instead of using an image given by, e.g., the manufacturer, is to preserve similarities (e.g., the same lighting conditions and resolution) between the reference image and the scene image.

A problem occurs when the product does not have a rectangular shape. In those cases, the reference image, when captured in the robot environment, will have segments

of the background included in the reference image. These segments create an unpredictable nature for the approach described in the following sections when evaluation a region consisting of the background. The solution was to crop out the background from the product manually.
When comparing similarities, if the reference image's pixel is transparent, the pixel is ignored and thereby not classified as cardboard. An image before and after removal of the background is shown in figure 3.4.

(a) The original reference image.

(b) The reference image with the background removed.

Figure 3.4: The figure shows the reference image with a non-rectangular shape. As shown in (a), the image contains part of the background. These areas are undesired as the solution approach is based upon comparing image regions against each other. The solution was to remove the background manually, as seen in (b).

3.3.2 Product Detection

The first step was to identify the product from the image captured by the camera. Because the camera captures the images in RGB-D, the point cloud of the image was stored for use in the creation of the collision map (section 3.4), while the 2D-RGB part was used for product detection and cardboard estimation.

The SIFT method (described in section 2.7) was used to identify the product in the scene image. The product detection solution was implemented with OpenCV and its tutorial [69]. If the product was identified, a bounding box containing the product was created (this was illustrated in figure 2.7).

When the product was identified, the image from the bounding box was aligned with the reference image by doing a perspective transformation. This transformation gave

the scene image the same dimensions and perspective as the reference image, as seen from figure 3.5. The transformation was accomplished with the function `warpPerspective()` from OpenCV. Having the images aligned made the pixels of the two images appear approximately in the same place, which was necessary for the cardboard estimation.

(a) Scene image cropped. (b) Scene image aligned. (c) Reference image.

Figure 3.5: The figure shows the result of the perspective transformation of the image (a) to the frame of image (c). The result is shown in (b).

3.3.3 Cardboard Detection

The approach for estimating the cardboard is based on measuring differences in the reference image and the scene image. If a region in the images differs, it was classified as cardboard. In practice, the estimated region could contain anything, but this approach goes under the assumption that the estimated occlusion is cardboard. The approach uses the HSV color space for comparing the differences. Thus, three new images were created for the reference image and the aligned image, one for each component (H, S, and V).

This approach is only estimating the cardboard, other possible collisions (e.g., with other products or the shelf) had to be taken into account. The solution for including other collisions will be described in section 3.4.

Smoothing

Before the images were evaluated, there could be much noise in the images due to, e.g., reflection. By iterating through the image with a window of a predefined size, setting all the pixels within the window to the mean value of the region, the result provided images more suitable for evaluation. The process is seen in figure 3.6.

(a) Scene image (RGB). (b) Scene image (only Hue). (c) Scene image (Hue) after window.

Figure 3.6: The figure shows the processing of the Hue image. Although the figure only displays the Hue image, the process is done for the Saturation and Value as well. Figure (a) displays the original (aligned) image, figure (b) displays the Hue channel of the original image, and finally, figure (c) displays the Hue after a window of 25 pixels.

Evaluation

As described in section 2.4, evaluating Hue does only make sense when the saturation is above some threshold. Therefore, depending on the values of H, S, and V, the evaluation was conducted on the component with the most dominant traits. In addition to the description in section 2.4 is the inclusion of saturation. If the saturation is low on, e.g., the scene image, and high on the reference image, this suggests that there is occlusion present. This is the case with figure 3.6(a), where the white cardboard covers parts of the yellow product. Therefore, if the difference of the saturation is above some threshold, it is classified as cardboard. A detailed description of the evaluation is in equation 3.1, where the different variables are described in table 3.2.

Table 3.2: Description of the variables use in the following section.

Variable	Description
Sat_{ref}	Saturation value of the reference image.
$\text{Sat}_{\text{scene}}$	Saturation value of the scene image.
$\text{th}_{\text{sat}}(\text{V})_{\text{ref}}$	Threshold value (saturation) of the reference image.
$\text{th}_{\text{sat}}(\text{V})_{\text{scene}}$	Threshold value (saturation) of the scene image.
Diff_{hue}	$\mid \text{Hue}_{\text{ref}} - \text{Hue}_{\text{scene}} \mid$
Diff_{sat}	$\mid \text{Sat}_{\text{ref}} - \text{Sat}_{\text{scene}} \mid$
Diff_{val}	$\mid \text{Val}_{\text{ref}} - \text{Val}_{\text{scene}} \mid$
V	V (brightness) from the HSV color space.
`THRESH_HUE`	Threshold value Hue.
`THRESH_SAT`	Threshold value Sat.
`THRESH_VAL`	Threshold value Val.

$$\text{Evaluate} = \begin{cases} \text{Saturation}, & \text{if} \quad \text{Diff}_{\text{sat}} \geq \text{Sat}_{\text{thres}} \\ \text{Hue}, & \text{elseif} \quad \text{Sat}_{\text{ref}} \geq \text{th}_{\text{sat}}(\text{V})_{\text{ref}} \\ & \text{or} \quad \text{Sat}_{\text{scene}} \geq \text{th}_{\text{sat}}(\text{V})_{\text{scene}} \\ \text{Value/Brightness}, & \text{else} \end{cases} \tag{3.1}$$

The component (H, S, or V) chosen to be most suitable for evaluation was determined by equation 3.1. From that component, the absolute difference between the reference image and the aligned image was computed. The classification was executed by checking the absolute difference against a threshold value. If the difference was higher, it was classified as cardboard, as seen in equation 3.2, where X is eighter Hue, Saturation, or Value.

$$\text{Classify pixel as} = \begin{cases} \text{Cardboard}, & \text{if} \quad \text{Diff}_X \geq \text{THRESH\_}X \\ \text{Product}, & \text{else} \end{cases} \tag{3.2}$$

Figure 3.7 shows the results after classifying the pixels. The resulting image does contain outliers that are undesirable when computing a robot path.

(a) Scene image. (b) Reference image. (c) The estimated cardboard.

Figure 3.7: The figure illustrates the cardboard estimation on the product Oboy. Figure (a) is the scene image, with a perspective transform to align with the reference image (b). The result after conducting the evaluation is displayed in (c).

Opening and Closing

The detection method described in the previous section will potentially produce misdetection as False Positives (FP) or False Negative (FN), as seen in figure 3.7(c).
By assuming that the only occlusion is cardboard, and that the cardboard and product will occupy continuous areas, it is possible to remove or fill the structural outliers. The

method used is morphological operations by opening and closing, described in section 2.5. The implementation was done with OpenCV, and the method preserves the overall structure of the estimate.

The chosen approach for opening and closing was to first to close the image (filling the holes) then continue by opening (removing outliers).
The process of closing and opening of the original classification is illustrated in figure 3.8.

(a) Original binary image.

(b) Binary image after closing operation.

(c) Binary image after opening operation.

Figure 3.8: The figure shows the morphological operation with closing and opening. The process is conducted on binary images, i.e., the illustration with the background is only meant for illustration purposes. Figure (a) display the original estimate, (b) display the image after closing, closing the gap between some of the pixels. Then, in (c), the opening removes some of the structural outliers. The window size of the structuring element is 2x the size of the moving window. From figure (c), a region of misclassification is shown.

As shown, the process is removing some of the outliers, but also creating a region of misclassifications. These misclassifications had to be removed as they could occlude the robot path. The problem was handled by only keeping the biggest contours.

Contours

Contours represent the outline of a shape. In this case, the outline of the white pixels in the binary image. By finding the contour with the largest area, removing everything else, the remaining cardboard estimation will be in one continuous area. However, if the estimated cardboard was detected in the wrong area, giving a lot of FP, the resulting estimate after morphological operations and contours can create a large misclassified area (this can be seen in figure C.6.1). The result after beneficial use of contours is seen in figure 3.9.

(a) Estimated cardboard after morphological operation.

(b) Estimated cardboard after keeping largest contours.

Figure 3.9: Figure (a) display the image after the morphological operation. Only keeping the largest contours, the resulting cardboard estimate is displayed in figure (b).

The result from figure 3.9(b) is transformed back to the original frame with the inverse `warpPerspective()` by additional the flag `WARP_INVERSE_MAP`, also done in OpenCV.

Figure 3.10: The figure displays the resulting cardboard estimation in front of the product, marked in green. As seen from the figure, the cardboard is only estimated in front of the detected product. With the estimated cardboard, this image is the basis for generating the collision map in section 3.4.

3.4 Proposed Solution for Collision Map

This section describes the proposed solution for generating the collision map. The collision map is created with OctoMap, which can use point cloud as input.

The collision map should contain all visible obstacles and account for areas not in

the FOV of the camera. The target product (the one the robot is about to pick) should be removed from the point cloud as it should not be included in the collision map.
The point clouds for representing the *Collision Point Cloud* consists of four point clouds, which will be described in detail in this section. Those are:

- **The *Product Point Cloud*:** used to represent the area within the point cloud that belongs to the product. The depth of the product (which is not available from the point cloud) was retrieved from the *Product File*.
- **The *Cardboard Point Cloud*:** because the *Product Point Cloud*'s area was removed from the *Collision Point Cloud*, part of the cardboard could be removed in the process. Therefore, the *Cardboard Point Cloud* was created and inserted after the processing of the point clouds.
- **The *Occlusion Point Cloud*:** this point cloud contains everything within the cameras FOV. Besides, because of the undefined area behind the FOV, this point cloud contains projected points into the depth axis to restrict movement in obscured areas.
- **The *Restriction Point Cloud*:** this point cloud was for restricting the robot's movement to be contained within the shelf, and not colliding with areas outside the FOV.

Illustrating images of the point clouds described is available in figure C.1, C.2, and C.3.

3.4.1 Product Point Cloud

From section 3.3.2, the product and the corresponding bounding box were estimated in 2D. By matching the pixels from the bounding box in the 2D image to the point cloud, the area contained by the product were found. Because the 2D image was created from the information in the point cloud, both images contain the same pixels, and looking up the 2D point in the point cloud is achieved by `point_cloud->at(x,y)`, where (x, y) is the pixel from the 2D image.

However, it was also necessary to know the pose [3] of the product. The rotation and translation of the product, relative to the camera's origin, were estimated by comparing the product's normal with the origin's normal. The normal to the product was calculated by finding the plane passing through the product's front. The plane was estimated by finding the best match of a plane from the SIFT-keypoints belonging to the product.

[3] Pose is the position and the orientation of the object [34, p. 585].

Because the keypoints did not necessarily correspond to the product of interest, all keypoints outside the bounding box were rejected. Illustration of the inliers and the rejected keypoints are shown in figure 3.11.

Figure 3.11: The figure shows the product of interest, marked with the green bounding box. The blue circles are the keypoint inliers, and the red circles are the keypoint outliers. This figure is cropped to only show information of interest.

The plane was created by making an optimal model of a plane, given a set of known inliers (SIFT-keypoints). The method used RANSAC as the estimator, described in section 2.3, and was implemented with inspiration from the PCL tutorial [70]. The method requires the point cloud along with the inliers as input. The output is the coefficients satisfying the equation of the plane given by equation 3.3 [71, p. 408].

$$ax + by + cz + d = 0 \tag{3.3}$$

Where a, b, c, d is some constant. From this equation, the normal to the plane is also provided by $v_{normal} = (a, b, c)$.
By comparing the normal vector of the plane to the origin vector $v_{origin} = (0, 0, 1)$, the rotation of the plane was calculated and gives the rotation in quaternions [4].

The translation from origin to the plane is just the normal vector $v_{normal} = (a, b, c)$, given by the coefficient from equation 3.3. The result is shown in figure 3.12, where a white bounding box is the estimate of the pose of the product. The size of the bounding box (width, height, and depth) was available from the *Product File*. Because of the inaccuracy of the camera (see section 3.1.3), the bounding box is 2cm larger in all directions.

[4] Quaternions are another way to represent rotation. It was essential for the code's implementation when using PCL, but it is not necessary for understanding this report. More information about quaternions is found in [72, chap. 5].

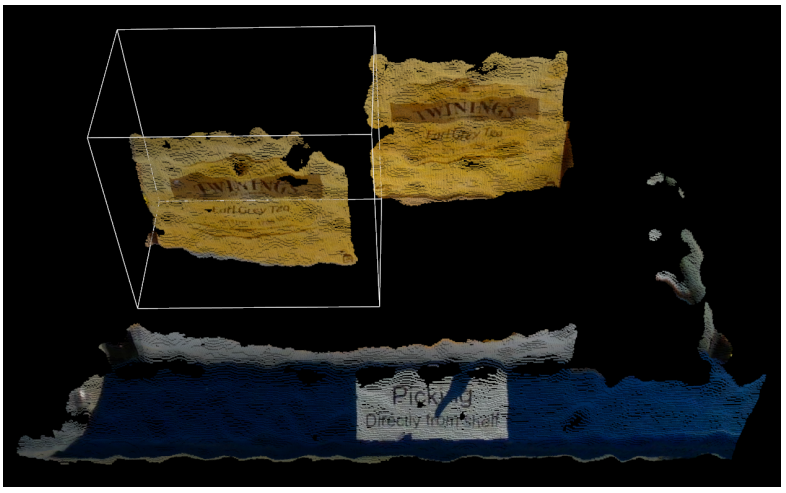
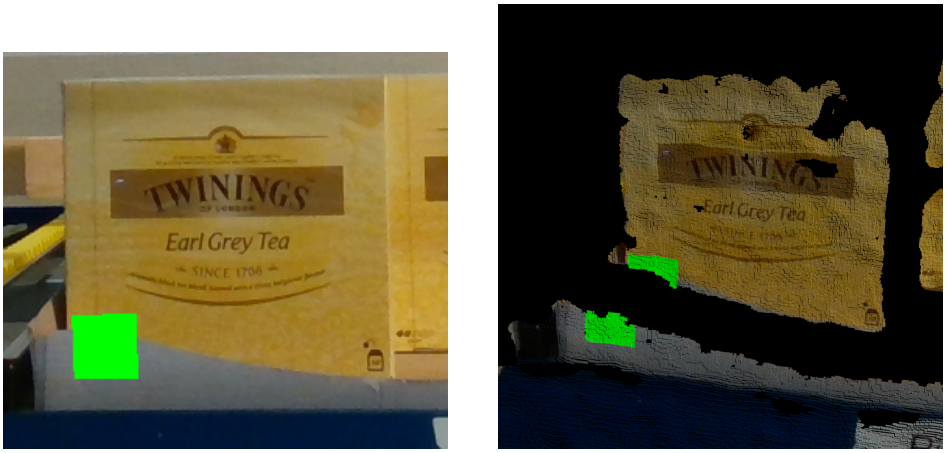


Figure 3.12: The figure displays the bounding box of the product, illustrated by the white cuboid.

3.4.2 Cardboard Point Cloud

The *Cardboard Point Cloud* contains the points in the point cloud belonging to the cardboard. These points were located by looking up the (x, y) coordinate from the binary image to the corresponding (x, y, z) point in the point cloud, as illustrated by figure 3.13.



(a) 2D Image with the estimated cardboard. (b) The cardboard estimated in the pointcloud.

Figure 3.13: The figures illustrate the estimated cardboard pixels in the point cloud, highlighted in green. As can be seen from (b), some of the cardboard's points in (a) are projected into the product. For illustration purposes, the image (a) is not a binary image, but the actual image of the scene with the estimated cardboard highlighted in green.

Remove Outliers from Cardboard Point Cloud

Because some pixels from the *Cardboard Point Cloud* does not necessarily belong to the cardboard, those points had to be removed, as these outliers will contribute to occupy

the area where the product resides.
The way to resolve this issue was to find the closest cardboard point and remove all points with a depth difference of more than 2cm. The reason was because of the assumption that the cardboard will be located in front of the scene. In the case where the product is adjacent to the cardboard (i.e., closer than 2cm), the pixels will not be removed. The solution to compensate if the points are included is described in section 3.4.6.

3.4.3 Restriction Point Cloud

Because the adjacent shelves and the shelf compartment located above were all outside the FOV, the robot path had to be restricted. A point cloud representing the area outside the FOV was created to restrict the movement to be within the product's shelf compartment. This *Restriction Point Cloud* consists of points for every 5mm around the workspace, assuming that the shelves are of known size.

3.4.4 Occlusion Point Cloud

The point cloud generated from the depth camera does not contain any information about the area behind the visible space. I.e., the area behind the products are not defined. When using the point cloud as the input for the collision map, all undefined spaces are interpreted as free space. If not dealing with the undefined areas, this could create collisions, as all concealed areas could contain obstacles. For this reason, the point cloud of the scene (the raw point cloud, not processed in any way) was projected into the depth for every 5cm, adding the new assumption that all unseen/occluded areas are collisions. However, the previously created bounding box of the product was used to remove the area defined by the product, because the product is not part of the collision map. The resulting point cloud is seen in figure 3.14.

3.4.5 Combining Point Clouds and Creating the Collision Map

With the point clouds created, they were combined to create the *Collision Point Cloud*. A color-coded version of figure 3.14 is in figure C.1, C.2, and C.3. The point cloud was published to ROS, and the resulting collision map is shown in figure 3.15, visualized in RViz.

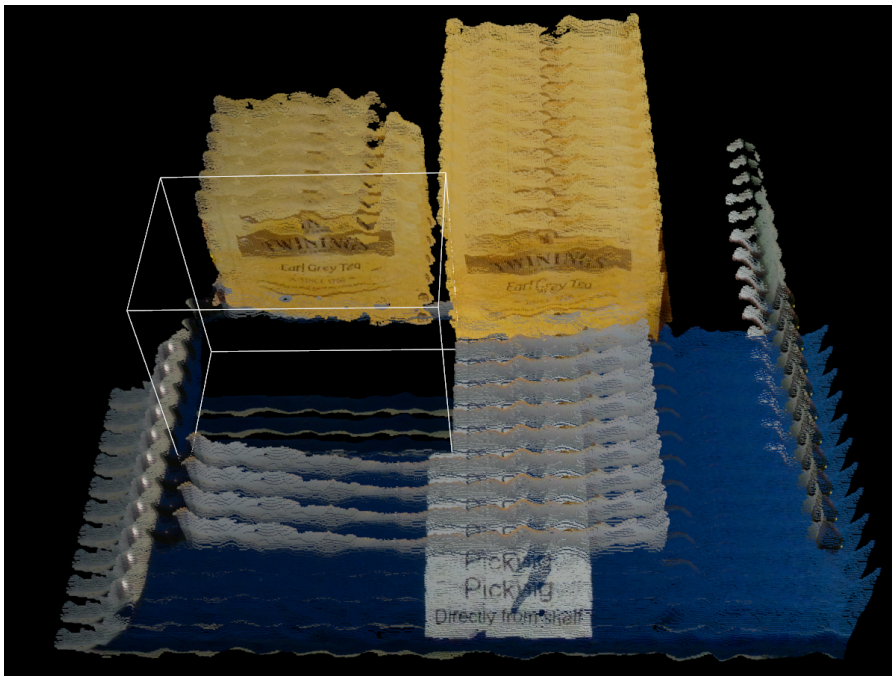


Figure 3.14: The figure displays the occlusion point cloud, with the product for retrieval cropped out, illustrated by the withe bounding box.

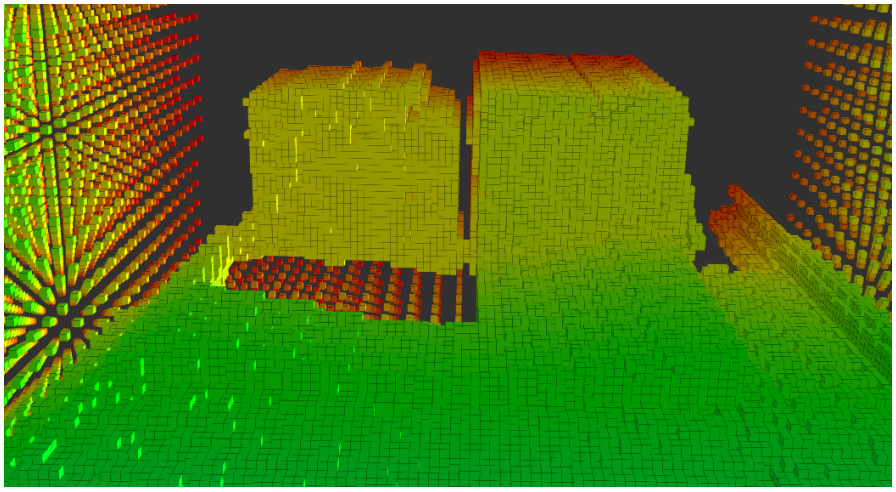


Figure 3.15: The figure displays the resulting OctoMap used for collision detection. The image is captured from RViz. The voxels are color-coded, with green representing voxels close to the camera, and red voxels further away.

3.4.6 Attaching the Product to the Robot

For the robot to retrieve the product, it had to navigate the provided path and check for collisions. Furthermore, the generated path had to be calculated concerning the attached product. In other words, if the product is not accounted for when generating a robot-path, it is likely to find a shorter path where the product collides with obstacles. This problem was solved by attaching a cuboid with the bounding box's dimensions

to the robot's tool-tip (suction cup) when it had reached the pick-point destination. However, due to inaccuracy and the problem mentioned in section 3.4.2 (when cardboard is adjacent to the product), the virtual product was reduced by 0.5cm in all directions. The robot with an attached product is seen in figure 3.16.

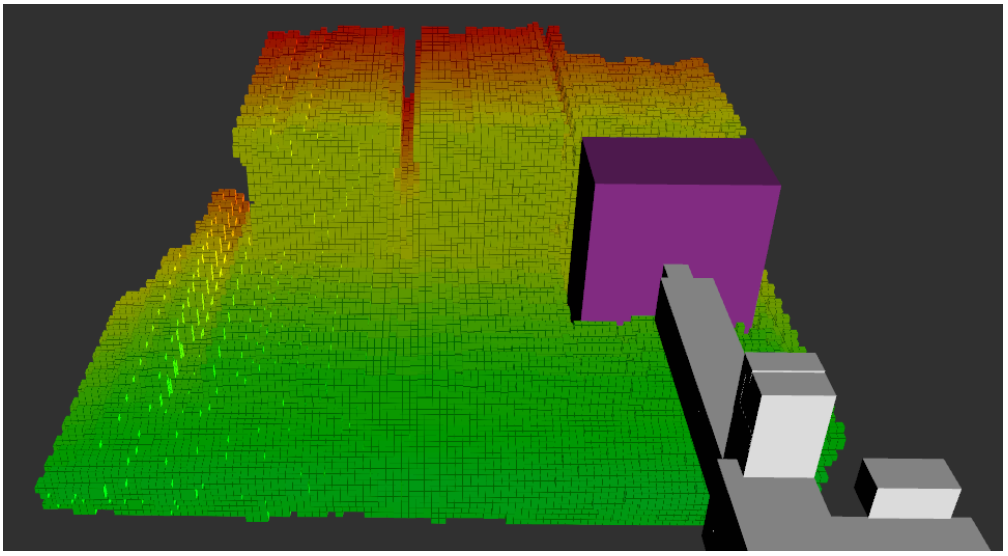
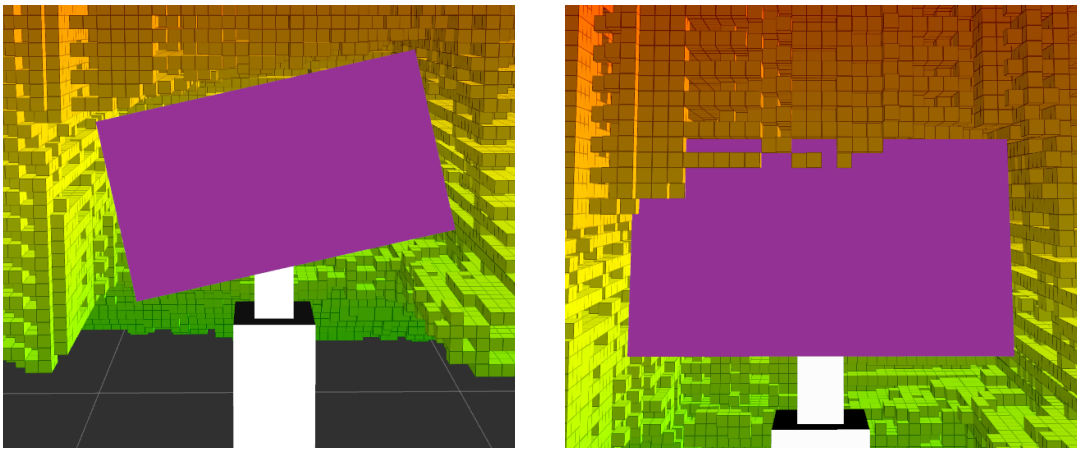


Figure 3.16: The figure displays the collision map in RViz. The purple cuboid is the virtual product attached to the robot. When the product is attached, it acts as part of the robot, and the robot path will not make the attached product collide with collisions.

3.4.7 Changes to Collision Map when Picking Skewed Products

When picking skewed products, several problems occur. Because the collision map and pose estimation of the product was created before the pick, the position and orientation of the physical product cannot be assumed to be the same after the robot has reached the pick point. This is because when the product is picked, it is attached perpendicular to the tool-tip. It might also be pushed to straighten and to achieve a sufficient vacuum, depending on the degree of orientation. This additional push might move the product into the estimated collision. The problem with the product being moved into the collision is illustrated in figure 3.17. Do note that the illustrated scene is staged, and the figure is only supposed to demonstrate the situation.



(a) The collision map in RViz with the product attached with the estimated pose.

(b) The collision map in RViz with the product attached perpendicular to the robot.

Figure 3.17: The pictures show the problem of picking a skewed product. The collision map was created considering the skewed product's pose, and the area contained by the product was removed, as seen in (a). When the robot moves to the product's pick-point, there is some additional pushing to achieve a sufficient vacuum, and the product is mounted perpendicular to the robot's tool-top, as seen in image (b). Not having the same pose of the product in the visualizer and on the real robot means that the collision map, which was estimated when the product was skewed, could be a false representation. The error in the pose could lead to the product being located within a collision, as illustrated in figure (b), where the top left corner is inside a collision and can lead to no viable path for retrieval.

The solution to picking skewed products is based on the assumption that the product's width is larger than the depth. Furthermore, the solution breaks with the previous assumption that all unseen areas are collisions. When removing the area contained by the product from the point cloud, the orientation is ignored, using the width instead of the depth. This creates a larger bounding box and means that the cropped area is large enough for the product to be oriented without being stuck in collisions.
The resulting collision map is shown in figure 3.18.

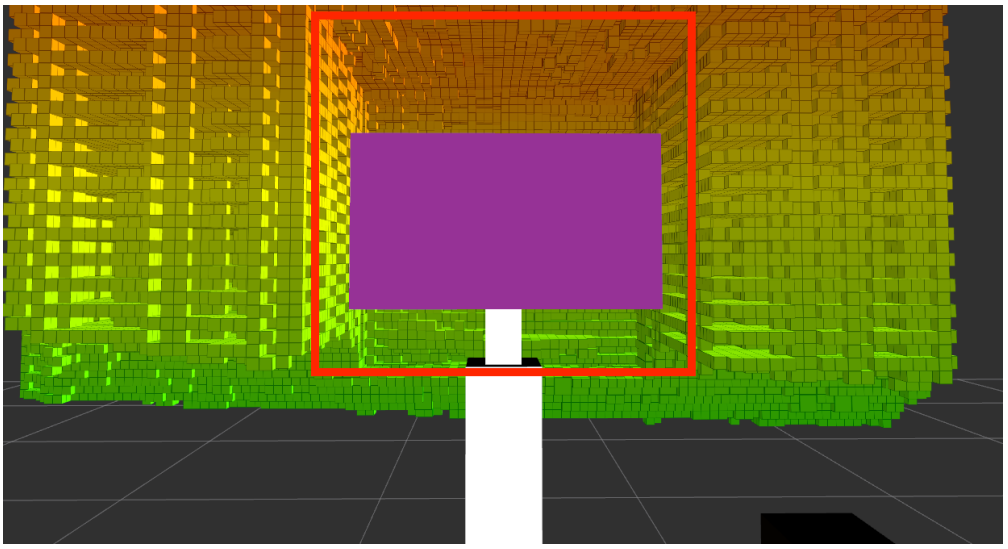


Figure 3.18: The figure display the final collision map in RViz. The red rectangle is the new cropped area for the product. The purple box is the virtual product, and is 0.5cm reduced in all directions.

The solution for picking skewed products is not optimal, and are discussed in chapter 5: Discussion and Future Directions. The reason for the implementation was to be able to test picking of skewed products.

3.5 Proposed Solution for Robot Path

The path planning algorithm used was the RRT-Connect, described in section 2.8. The purpose of the solution in this section was to generate a robot path to the pick-point (i.e., target location), attach a virtual product to the robot (as explained in section 3.4.6), and retrieve the product back to the initial position.

3.5.1 Generating Pick-Point

The pick-point for each product has a position and rotation. In this approach, the pick-point was chosen to the center, and with the same orientation as the product.
However, as described in section 1.3, the end-effector can not rotate around the Y-axis. Meaning that pick-points with an orientation around the Y-axis will be unreachable for the end-effector. This is the case if, e.g., the product is skewed.
The solution for this was setting the orientation around the Y-axis to 0°, regardless of the degree of orientation.

3.5.2 Cartesian Control for Picking Skewed Product

The proposed solution for picking skewed products was with the use of Cartesian control. There were two primary reasons for this. The first was to test whether a Cartesian control was sufficient for picking products. The second reason was for the predictability of a Cartesian control compared to the RRT based planners. With Cartesian control, the undesired random movement within the distribution container is removed. However, there are some new uncertainties in the collision map when picking skewed products, as mention in 3.4.7.

A list of waypoints is used to implement the Cartesian path in MoveIt!. The robot moves in a linear path between each waypoint, and if collisions are detected, the path fails. The idea was to use the point cloud and the available data in the *Product File* to calculate waypoints for retrieving the product without collisions.
Two waypoints are needed to extract the product from the distribution container. The first waypoint is needed to move to the product above the cardboard, while the second waypoint moves the product outside the shelf.

There are several ways to implement such a solution. One way, assuming the cardboard is covering the lower region of the products, is explained below:
This solution worked on the products in the first row, where the cardboard was adequately estimated. The distance needed to lift the product above the cardboard (Waypoint 1) equals the height of the cardboard (denoted h) + some safety distance (denoted ϵ). This is illustrated in figure 3.19.
The distance needed to move the product out of the shelf (Waypoint 2) equals the pick-point's **D**istance to the **C**ardboard (denoted Z_{DC}) + the **P**roduct's **D**epth (denoted Z_{PD}) + ϵ. Illustrated in figure 3.20.

However, because the front cardboard is not visible for all the rows, and therefore not necessarily estimated, the mentioned approach needs some restructuring to work sufficient for all the products. The problem is illustrated in figure C.11. A suggested workaround is to estimate the cardboard in the first row for all the products and save the estimated *Cardboard Point Cloud* to use on the rest of the products.

The implemented solution used in the experiment is a simplification of the approach mentioned above. Waypoint 1 was created with a fixed height of 10cm above the pick-point. This ensures that the product is above collisions, as no cardboard extended for more than 10cm. For the retraction, Waypoint 2 was created a fixed safe distance away from the shelf ($\approx$ 20cm outside the shelf).

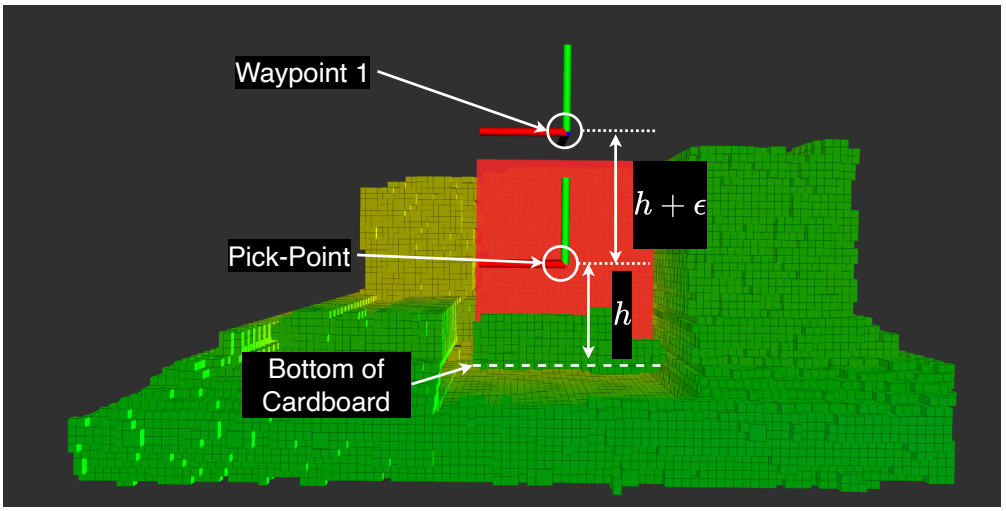


Figure 3.19: The figure illustrates the calculation of Waypoint 1, the upward distance in Y-direction. h is the distance from the pick-point to the bottom of the product marked in red. ϵ is the buffer distance to ensure clearance to the cardboard.

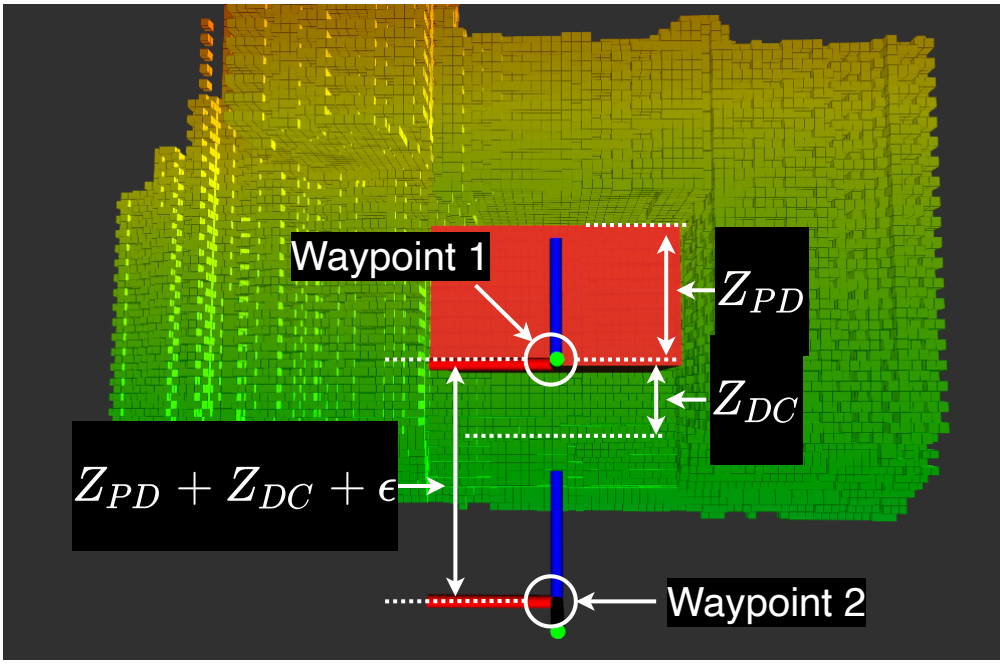


Figure 3.20: The figure illustrates the calculation of Waypoint 2, the retraction in Z-direction. The Z_{PD} is the product's depth, and Z_{DC} is the distance from the pick-point to the cardboard. The retraction in the Z-direction is $Z_{PD} + Z_{DC} + \epsilon$.

Chapter 4

Experimental Evaluation

In this chapter, the conducted experiments and corresponding results are presented. Two experiments were conducted, "Picking from Distribution Containers" and "Picking Skewed Products."

4.1 Evaluation Metrics

The evaluation metric used in this thesis is presented in this section.

4.1.1 True Positive, True Negative, False Positive, False Negative

The terms "True Positive (TP)," "True Negative (TN)," "False Positive (FP)," and "False Negative (FN)," are used in the classification and for calculating the score of the other evaluation metrics. Figure 4.1 explains the use of the different terms.

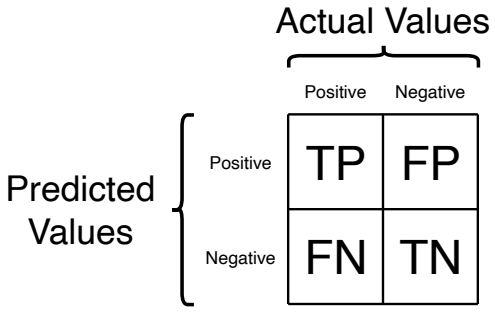


Figure 4.1: The figure displays the use of the terms "True Positive," "True Negative," "False Positive," and "False Negative."

4.1.2 Ground Truth: Segmentation of the Images

In both the conducted experiments, the ground truth images were manually segmented, and created with Labelbox. The output from Labelbox, after the segmentation, was a binary image with white pixels representing the object of interest, and everything else as black pixels, this is seen in figure C.4.
When segmenting the product, it could be partly occluded (by, e.g., the cardboard). The segmentation had to be "guessed" in the areas the product was not visible. This leads to some error-margin as the ground truth does not necessarily become perfect.

The same approach was used for the cardboard segmentation, but with one difference, the ground truth image was manually labeled from the predicted bounding box of the scene image. If the SIFT method failed to identify the product, the resulting bounding box would be distorted, and the segmentation would yield no cardboard visible. This is seen in the product tea05 (figure C.6) and tea06 (figure C.7).

4.1.3 Intersection over Union

The Intersection over Union (IoU), is illustrated by 4.2. It is the area of overlap divided by the union and is one of the most used metrics in object detection [73].

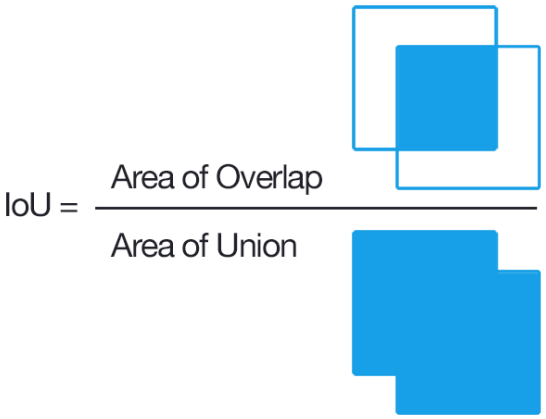


Figure 4.2: The figure illustrates the IoU metric. Two bounding boxes are evaluated on the degree of overlap, and the intersection is divided by the union. If the two bounding boxes are correctly aligned, a score of 1 or 100% is given. `The figure is reprinted in unaltered form from Wikimedia commons, File:Intersection over Union - visual equation.png, licensed under CC BY-SA 4.0.`

The IoU was used slightly different when evaluating the performance of product detection and cardboard estimation. The two different IoU implementations are described below.

Evaluation Product Detection

In this scenario, it is only the product that is of interest. The IoU is calculated as in equation 4.1, where ϵ is a small number to avoid potential division by zero if there is no product in the scene. The IoU for evaluation of the product is denoted IoU_{PD} in this thesis, "PD" for Product Detection.

$$\text{IoU}_{\text{PD}} = \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP} + \epsilon} \tag{4.1}$$

A real example of this is seen in figure 4.3, where the product oboy got an IoU_{PD} of 92.84%.

Figure 4.3: The image illustrates the product detection of the product "oboy01." The green bounding box is the estimated region of the product. The blue and red circles are not relevant to the image.

Evaluation Cardboard Detection

There are two cases for every pixel when using IoU for evaluating the performance of cardboard detection. Either, the pixel represents cardboard, or it represents "no cardboard." When evaluating the cardboard, both the TP (the pixel representing the cardboard) and the TN (the pixel representing the background) is essential to classify. If using the IoU for evaluating the performance of the cardboard alone, it will yield 0% for all cases where there is no cardboard in the image, because there will be no intersection. The IoU for evaluating the cardboard is denoted IoU_{CD}, "CD" for Cardboard Detection.

To make the IoU_{CD} score account for the problem when there is no occlusion, the IoU is calculated separately for the cardboard (denoted $\text{IoU}_{\text{cardboard}}$) and the background (denoted $\text{IoU}_{\text{background}}$) before taking the mean, i.e., the mean IoU. When evaluating the background, the classification is "inverted," giving TP to the background and FN to the

cardboard.

$$\text{IoU}_{\text{cardboard}} = \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP} + \epsilon} \quad \text{True class is Cardboard.} \tag{4.2}$$

$$\text{IoU}_{\text{background}} = \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP} + \epsilon} \quad \text{True class is Background.} \tag{4.3}$$

$$\text{IoU}_{\text{CD}} = \frac{\text{IoU}_{\text{cardboard}} + \text{IoU}_{\text{background}}}{2} \quad \text{Mean IoU combining the classes.} \tag{4.4}$$

4.1.4 Accuracy

The Accuracy metric is shown in equation 4.5, as implemented in this thesis.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.5}$$

The Accuracy gives information about the proportion of the data that was correctly classified. One drawback of the Accuracy metric is when there is no cardboard in the scene. If there is no cardboard, and the estimation yield "no-cardboard," the Accuracy will be 100%. Furthermore, since the cardboard in most cases only covers a small proportion of the image, if the prediction never estimates any cardboard, the Accuracy would still be very high. Therefore, the Accuracy metric is not suitable for evaluating the classifier's performance, but rather to see the overall proportion of correctly classified pixels, as this is useful when evaluating if the detection is "good enough" to use to with the collision map.

4.1.5 Succsess Rate

For robot path planning, there are only two results. Either the path planning succeeds, or it fails. The success rate is the number of successful experiments divided by the number of conducted experiments.

4.2 Table Overview

In this section, detailed information of the header abbreviations of the tables and the information it contains is described.

Position (Pos.): The row within the distribution pack the product was located.

ID: The identification of the product. The ID is unique within each experiment.

Category: For each pick, the product was categorized based on the "reason for failure." The different terms are described here:

- **Target:** The pick failed and did not manage to find a path to the target (pick-point).
- **Home:** The robot picked the product, but did not find a path back to the initial state.
- **Collision:** The product collided during the extraction.
- **Success:** The pick was successful.
- **Not reachable:** The product was out of reach.

Some additional information regarding collisions: Collisions are allowed, but only to some extent, divided into "soft-collisions" and "hard-collision." Those terms are not used in the tables, but all collisions resulting in an unsuccessful pick is a "hard-collision." The distinction between the two terms used in this thesis is whether or not they manipulate the workspace, i.e., if the initial position of the setup is changed. If a product collides with the cardboard, but are still successfully extracted, this would be a successful pick but with a "soft-collision." Due to the difficulties in objectively determine all "soft-collisions," the term is not used in the tables with results, resulting in the collision category only consisting of "hard-collisions."

4.3 Experiment: Picking from Distribution Containers

The purpose of this experiment was to evaluate the performance of the proposed solution under simplified environments. E.g., everything is set back to the initial state before each pick. Using data gathered from this simplified experiment allows for testing in edge-case scenarios, detecting the limits and constraints of the system. Testing in both a "normal" situation and in an "edge-case" setting can give insight about where the system falls short or needs improvements.

The Procedure of the Experiment

- The reference image was captured under the same environment and light as the conducted experiment.

- The success rate on the picking was evaluated both in simulation and on the physical robot, denoted "SR Sim." and "SR. Rob." correspondingly.
- If a pick failed, the product was manually removed from the distribution package.
- A pick was classified as "failed" if no valid robot path was found, or if there was a "hard-collision."
- Everything was set back to the initial state if there were any collisions or movement within the distribution package.
- Products that were not reachable, because of limitation in the reach of the end-effector, does not count in on the success rate on the robot path planning.

4.3.1 Experimental Setup

In this section, the setup of the experiment is described. The experiment contained eight different product types. The product types were chosen from a pallet of groceries that fulfilled the assumptions and criteria from section 1.4. Those products, with the corresponding reference images and images captured from the robot, are available in Appendix A. A total of 88 products were evaluated for cardboard detection. In contrast, the path planner evaluation only contained 38 products because the end-effector did not manage to reach within all the rows of the distribution pack. The planner used was the RRT-Connect, described in section 2.8.1.

Figure 4.4: The figure displays the setup of the experiment. In the figure, one product have already been picked by the robot. All the products are in initial states.

Experimental Settings

Table 4.1: The table displays the image processing settings used in this experiment. "Morph. op." refers to morphological operation.

Setting	Value	Unit
Threshold hue	35	$\in [0-255]$
Threshold saturation	40	$\in [0-255]$
Threshold value	50	$\in [0-255]$
Window smoothing	25	pixels
Window morph. op.	55	pixels

4.3.2 Experimental Results

Here, the results of the conducted experiment are presented. Table 4.2 displays the average score for each product type.

Table 4.2: The table shows the average score for each product type. The data is from the experiment "Picking from Distribution Containers," with all the 88 products.

Product	TP	TN	FN	FP	Accuracy	IoU_{CD}	IoU_{PD}
oboy	11.66%	84.07%	3.02%	1.26%	95.73%	75.15%	90.58%
granola	3.14%	80.51%	2.28%	14.07%	83.65%	53.54%	94.78%
juice	4.13%	82.75%	5.30%	7.82%	86.88%	52.01%	93.11%
tea	4.09%	72.80%	2.89%	20.22%	76.88%	48.79%	73.53%
blenda	8.38%	75.85%	1.90%	13.88%	84.23%	65.60%	78.71%
asana	0.00%	76.60%	9.02%	14.39%	76.60%	38.30%	34.77%
kvikklunch	17.13%	32.39%	7.75%	42.73%	49.52%	33.84%	72.15%
sun	0.00%	61.15%	6.78%	32.06%	61.15%	30.58%	61.11%

Due to the cardboard location in front of the distribution pack, it was worth evaluate the product types based only on the first row. This is shown in table 4.3.

Table 4.4 displays the reason for failure, along with the statistics from the cardboard estimation.

Table 4.5 list the overall success rate of the product types along with the number of products evaluated (not all 88 products were used due to the limiting reach of the end-effector).

Table 4.3: The table shows the evaluation of the product types when only considering the first row ("Row 1"). The percentage is the average of all the products within each distribution pack in the first row. The data is from the experiment, "Picking from Distribution Containers."

Product	**Accuracy**	IoU_{CD}	IoU_{PD}
oboy	99.04%	97.52%	92.19%
granola	99.26%	96.87%	91.83%
juice	96.24%	90.05%	86.68%
tea	92.44%	78.61%	91.14%
blenda	91.93%	72.23%	85.46%
asana	68.75%	34.37%	45.42%
kvikklunch	65.43%	43.77%	86.06%
sun	60.75%	30.37%	54.92%

Table 4.4: The table lists the different categories along with the statistic from the cardboard estimation. For "Success," this means that 19 products had a successful extraction. The Accuracy, IoU, and FP are also included, which is the average value of the products within each each category. The category "Not Reachable" is not supplemented to the table as it provides no useful information when evaluating "Reason for Failure." The data in this table is from the experiment, "Picking from Distribution Containers."

Category	**Num. Prod**	**Accuracy**	IoU_{CD}	IoU_{PD}	**FP**
Target	4	55.62%	27.81%	2.30%	44.38%
Home	8	71.47%	35.74%	86.65%	16.41%
Collision	6	86.95%	57.90%	88.90%	0.41%
Succsess	19	93.58%	73.22%	90.47%	1.01%

Table 4.5: The table lists the success rate for the products used in this experiment, with the simulator's success rate (SR Sim.) and the success rate on the physical robot (SR Rob.). Along is the statistics from the cardboard evaluation. The data is from the experiment, "Picking from Distribution Containers."

Product	**Num. Prod.**	**SR Sim.**	**SR Rob.**	IoU_{PD}	**FP**	**Accuracy**
oboy	6	100.00%	100.00%	92.83%	1.38%	96.95%
granola	4	100.00%	75.00%	93.57%	0.84%	96.05%
juice	4	100.00%	75.00%	90.67%	0.84%	90.84%
tea	4	100.00%	100.00%	91.83%	1.66%	92.93%
blenda	4	100.00%	75.00%	86.08%	0.70%	94.58%
asana	4	25.00%	25.00%	67.34%	15.95%	74.73%
kvikklunch	4	100.00%	25.00%	80.01%	1.21%	80.29%
sun	9	0.00%	0.00%	56.83%	30.20%	61.99%

4.4 Experiment: Picking Skewed Products

This experiment was performed to explore the possibilities of picking skewed products. Picking skewed products are of importance since the products can not be assumed to be stationary. If a collision within the distribution container occurs, the product might move.

The Procedure of the Experiment

- The reference image was captured by the same camera.
- The skewed product was always rotated anti-clockwise when looking from the camera's position. As seen in figure 4.5.
- All products have to stand upright and have a depth > 5 cm.
- A pick was classified as "failed" if the robot path failed to generate a valid path, or if there was a "hard-collision."
- Only the reachable products were evaluated, i.e., the first two rows of the products in this experiment.
- The product of retrieval was skewed, with the right side of the product moved 3 cm back in the scene.
- All other products, except for the product for retrieval, was turned backward, to force the product detector only to detect the skewed product (because SIFT uses a reference image of the front of the product).
- Cartesian control was used to pick the products instead of RRT-Connect.

4.4.1 Experimental Setup

The products used in this experiment are the ones with an $\text{IoU}_{\text{PD}} > 80\%$ and Accuracy $> 80\%$ from the first experiment from table 4.5, namely "oboy," "granola," "juice," "blenda," and "tea." The experiment was conducted on the first two rows, providing a total of 22 products. The product "Kvikklunch" was not included in the experiment because the depth (surface the product balances) is 2.4cm and will not be suitable due to the experiment's procedure.

Figure 4.5: The figure displays the setup of the experiment. One product was skewed while the rest was in the original position. Because the skewed product occupies a larger area, the row in the back was removed. The other products were turned backward to force product detection to detect the skewed product.

Experimental Settings

Table 4.6: The table displays the image processing settings used in this experiment. "Morph. op." refers to morphological operation.

Setting	Value	Unit
Threshold hue	35	$\in [0 - 255]$
Threshold saturation	40	$\in [0 - 255]$
Threshold value	50	$\in [0 - 255]$
Window smoothing	25	pixels
Window morph. op.	55	pixels

4.4.2 Experimental Results

The tables evaluating the experiment are listed below in table 4.7, 4.8, and 4.9.

Table 4.7: The table shows the average score for each product type examining the cardboard evaluation. The data is from the experiment, "Picking Skewed Products."

Product	TP	TN	FN	FP	Accuracy	IoU_{CD}	IoU_{PD}
oboy	15.77%	76.88%	3.16%	4.19%	92.65%	82.34%	90.59%
granola	3.76%	89.38%	4.77%	2.09%	93.14%	61.82%	95.34%
juice	10.36%	81.85%	5.59%	2.21%	92.21%	68.46%	88.13%
tea	5.66%	64.02%	4.76%	25.56%	69.68%	53.43%	69.96%
blenda	11.39%	83.61%	2.03%	2.97%	95.00%	84.00%	70.73%

Table 4.8: The table lists the different categories along with the statistic from the cardboard estimation. For "Success," this means that 19 products had a successful extraction. The Accuracy, IoU, and FP are also included, which is the average value of the products in the experiment. The category "Not Reachable" is not supplemented to the table as it provides no useful information when evaluating "Reason for Failure." The data in this table is from the experiment, "Picking Skewed Products."

Category	**Num. Prod**	**Accuracy**	IoU_{CD}	IoU_{PD}	**FP**
Target	1	0.00%	0.00%	3.79%	100.00%
Home	2	78.47%	39.23%	91.93%	11.11%
Collision	0	N/A	N/A	N/A	N/A
Succsess	19	94.48%	76.68%	87.46%	1.72%

Table 4.9: The table lists the success rate for the products used in this experiment, with the simulator's success rate (SR) and the success rate (SR) on the physical robot. Along is the statistics from the cardboard evaluation. The data is from the experiment, "Picking Skewed Products."

Product	**Num. Prod.**	**SR Sim.**	**SR Robot**	IoU_{PD}	**FP**	**Accuracy**
oboy	6	83.33%	83.33%	90.59%	4.19%	92.65%
granola	4	75.00%	75.00%	95.34%	2.09%	93.14%
juice	4	100.00%	100.00%	88.13%	2.21%	92.21%
tea	4	75.00%	75.00%	69.96%	25.56%	69.68%
blenda	4	100.00%	100.00%	70.73%	2.97%	95.00%

4.5 Comparing Experiment

In this section, an excerpt from the statistics in the two experiments is used. The term "Normal" is referring to products used in the experiments "Picking from Distribution Packages," and "Skewed" is referring to the "Picking Skewed Products" experiments, in order to shorten the names.

Because the two experiments did not have the same number of product types, only the products used in both experiments are compared, and only the first two rows of each distribution pack. This means that only "Oboy," "Granola," Juice," "Blenda," and "Tea" have been compared.

Table 4.10: The table compares the average IoU_{PD} from the two experiments. "Normal" is referring to the experiment "Picking for Distribution Containers" and "Skewed," referring to "Picking Skewed Products." Only the two first rows of the distribution container are evaluated.

Product	IoU_{PD} **Normal**	IoU_{PD} **Skewed**	**Accuracy Normal**	**Accuracy Skewed**
oboy	92.83%	90.59%	96.95%	92.65%
granola	93.57%	95.34%	96.05%	93.14%
juice	90.67%	88.13%	90.84%	92.21%
tea	91.83%	69.96%	92.93%	69.68%
blenda	86.08%	70.73%	94.58%	95.00%

Table 4.11: The table list the categorizes for failure or success for the two experiments, with the number of products placed within each category. FP is the average False Positive of the products in both experiments.

Category	**Normal**	**Skewed**	**FP**	IoU_{PD}	**Accuracy**
Target	0	1	100.00%	3.79%	0.00%
Home	0	2	11.11%	91.93%	78.47%
Collision	3	0	0.09%	92.15%	89.92%
Succsess	18	19	1.47%	89.15%	94.61%

Table 4.12: The products that failed from table 4.11 are listed in this table with the individual product and the statistics. "Type" is referring to the experiment type, and the ID is the identification of the product. The "Normal" products all failed due to "Collision," and the "Skewed" products failed either because of "Home" or "Target."

Type	**ID**	**TP**	**TN**	**FN**	**FP**	**Accuracy**	IoU_{PD}
Normal	granola04	0.00%	92.01%	7.99%	0.00%	92.01%	95.89%
Normal	juice03	0.00%	82.63%	17.37%	0.00%	82.63%	93.67%
Normal	blenda02	12.54%	82.58%	4.59%	0.28%	95.12%	86.88%
-	-	-	-	-	-	-	-
Skewed	oboy06	0.00%	69.27%	13.98%	16.74%	69.27%	88.98%
Skewed	granola03	0.00%	87.66%	6.87%	5.47%	87.66%	94.87%
Skewed	tea04	0.00%	0.00%	0.00%	100.00%	0.00%	3.79%

Table 4.13: The table displays the success rate of the two experiments, for both the simulator and the physical robot. Information about the Accuracy and IoU_{PD} is supplemented for comparison. The products used in the evaluation was "Tea," "Juice," "Oboy," "Granola," and "Blenda," on the first two rows.

Experiment	**Simulator**	**Robot**	IoU_{PD}	**Accuracy**
Normal	100.00%	85.00%	91.00%	94.27%
Skewed	86.67%	86.67%	82.95%	88.53%

Chapter 5

Discussion and Future Directions

5.1 Discussion

This section discusses the result from section 4: Experimental Evaluation, followed by suggested further directions. This section also answers the main thesis' questions from section 1.4.

5.1.1 Product and Cardboard Detection

For some of the products, the image from the bounding box of the product is perspective distorted compared to the reference image. This distortion is seen by looking at "tea05" and "tea06" in figure C.6 and C.7, with their classification results from table B.2. The products have an IoU_{PD} of around 3% and 5%, respectively, resulting in an unreliable cardboard estimation. Because of the approach used for evaluating the cardboard, it is essential to have a high precision bounding box of the product, i.e., high IoU_{PD}. In this approach, because the cardboard estimation is based upon the bounding box of the product, the IoU_{PD} should be closer to, or above, 80% to yield a sufficient estimate for evaluating the cardboard. The reason for not proposing the estimate to be above 90% is because of the error-margin in the ground truth images, as described in section 4.1.2.

The cardboard will only occupy the outer edges of the distribution container. If the cardboard estimation is "flawlessly" estimated on the first row of products, it is not necessary on the rest of the products, because the cardboard estimation has already been performed on the first row. The result when only evaluating the first row is shown in table 4.3, with all product performing with an Accuracy $> 90\%$ and $\text{IoU}_{\text{PD}} > 80\%$, except "Asana," "Kvikklunch," and "Sun." Only evaluating the first row will also benefit

the cardboard estimation, as the cardboard is most prominent in the first row, and the degree of presence varies in the other rows because of the position of the camera compared to the products, as seen in figure C.11.

Comparing the product detection and cardboard estimation of the two experiments from table 4.10, the IoU_{PD} on "tea" and "blenda" decreased with about 15% and 20% on the "Skewed" experiment, lowering the IoU_{PD} to $< 80\%$, and consequently, categorizing the product as unsuitable for picking when skewed.

5.1.2 Robot Path Planning

This section discusses the results with the use of RRT-Connect and Cartesian control, as well as the cardboard estimation's influence on the generated path.

Evaluating the robot path planning from table 4.4, it is natural that when the $\text{IoU}_{\text{PD}} = 2.30\%$ (in category "Target"), there is no valid robot path to the target/pick-point since the product was not correctly identified. Looking at table 4.5, in the column "SR Sim.", "Asana" and "Sun" succeeded in only 25% and 0%, respectively, whereas the rest had an "SR.Sim" of 100%. Those two products also have an $\text{IoU}_{\text{PD}} < 70\%$, and, partly due to the low IoU_{PD}, the two products have FP of about 16% and 30%, compared to the successful products with FP about 1%. A high number of FP indicates that the cardboard estimation after the post-processing with opening, closing, and contours, is located in the wrong area, thus restricting the generation of a valid path, as seen in figure C.8 and C.9.

The RRT-Connect is time-consuming and uses unnecessary long trajectories, although not directly measured in the experiment. Looking at the video recording from one of the picks with RRT-Connect, [1, File: normal_granola.mp4, ID: granola04], the collision is due to a trajectory into the side of the cardboard tray. The RRT-Connect does have randomness involved when calculating the robot path (randomness from the sampling with exploring trees). This randomness, if combined with uncertainty or allowing for error-margin in the estimation, increases the possibility for collisions.

Cartesian control was used in the experiment with skewed products. The results from table 4.8 show that there were no collisions with the use of Cartesian control. Furthermore, Cartesian control used a calculation time $<$ 1sec in the conducted experiment, compared to RRT-Connect, which from the video, [1, File: normal_granola.mp4, ID: granola01], used a time of $\approx$ 30sec. Because the Cartesian control has linear paths, and the path

is a list of determined waypoints, there is no randomness in the same manner as with RRT-Connect. The drawback is that it will not find any other path if the waypoints are unreachable.

There is a noticeable difference between the two experiments when examining table 4.11 and the "Categories." The "Normal" products failed due to "Collision," with FP $\approx 0\%$, meaning they had a "hard-collision" in the path back to the initial state. In contrast, the "Skewed" products with Cartesian control had no collision. Instead, they all failed due to "Target" or "Home," i.e., there was no valid path to retrieve the products. With the "Target" having an FP $= 100\%$, there was naturally no valid path to the target, while for "Home," the FP is $\approx 11\%$ comparing to $\approx 1\%$ of "Success" and "Collision." The "Home" category indicates that the classified region of cardboard is in the wrong area.

The overall success rate from the two experiments is displayed in table 4.13. The "Normal" experiment did have $\approx 10\%$ higher Accuracy and IoU_{PD} than the "Skewed" experiment. However, worth noticing is that the "Skewed" experiment with Cartesian control succeeded with all the products that were successful in the simulator. This consistency is valuable as the simulator represents the robot's interpretation of the world. The SR on the physical robot should be as close as possible to the simulator's SR.

5.1.3 Picking Skewed Products

The solution for enabling picking of skewed products was to remove a more substantial area around the product in the collision map. This allowed for pushing while the product remained outside the collision map. The approach introduced uncertainties that need to be examined to determine if the approach is a safe and reliable solution. Safety is an issue because the approach assumes that there is no collision when pushing the product. This can not always be assumed if the products can move within the distribution container. That said, the robot path when allowing for pushing, succeeded 19 out of 22 times and did not have any "hard-collisions," indicating that the approach is worth further exploring.

5.2 Future Work

This section suggests future work and direction based on the results and research done in this thesis.

5.2.1 Product and Cardboard Detection

The main challenge with generating a collision map in this thesis was the estimation of the cardboard. Estimating cardboard by comparing a reference image to a captured image of the products are sensitive to adjustments like light and shadows. This was compensated by capturing the reference image in the same environment as the conducted experiment. It is interesting to look into ways to use images received from third parties, e.g., the manufacturer. However, these images are not captured in the same environment as the system and will have increased color differences compared to the scene image. The light has a severe effect on the performance of the product detector (SIFT) and the cardboard estimation. If possible, introducing some flat light to brighten the scene without creating reflection can improve the result of the low-feature products.

The pose estimation method used in this thesis assumes a flat surface of the product. If the product does not have a flat front surface, the estimated pose will be unreliable. Pose estimating was part of the work done by Skutvik [74] on the same system in his master's thesis. However, the result was not conclusive, but the thesis digs deeper into pose estimation techniques, which have not been a significant part of this project.

Suggested future work for the estimation of cardboard is diverging in two directions. Either continue with primary color differences, creating an estimation method that works on a defined area of products, and maybe using machine learning to optimize the parameters. If only evaluating the first row and assuming that there is a cardboard present, it is possible to eliminate the variables by determining the threshold for classification by using, e.g., Otsu's method. Otsu's method selects the threshold value that maximizes the classes' separability by evaluating the histogram [75, p. 541]. However, the drawback and one of the reasons for not being used in this solution is because it separates the data into two classes, whether or not they exist [75, p. 545].

The other direction is looking into deep neural networks. With the machine learning segmentation algorithm like Mask R-CNN, as mentioned in section 2.2, one possible approach for further work is to use a segmentation based product detector to locate the product, and assume that everything else is collisions. Detecting and segmenting a product instead of the cardboard will probably be easier as one can rely upon similar products. One of the challenges with Mask R-CNN, as with most machine learning techniques, is that it requires a lot of labeled data.

5.2.2 Robot Path Planning

The following analysis and suggestion are only focusing on path planning and assumes that the cardboard, occlusion, and collision detection is of no concern.
It is interesting to investigate the performance of different planners, e.g., BIT*, RRT-Connect, and Cartesian control, when introduced to an unpredictable environment. I.e., if the cardboard shape is more complex, where a linear path (up and back) is not sufficient to retrieve the products. If continuing with experiments on different planners and more complex cardboard shapes, it is valuable to evaluate the planners on calculation time as well as the length of the path.
Reconstructing the program to estimate the cardboard in the first row and saving it to use in the collision map looks promising for future work if continuing with the same assumptions as in the conducted experiments. Furthermore, as mentioned in section 3.5.2, it is well suited for Cartesian control.

5.2.3 Picking Skewed Products

It is suggested to consider modifications to the system in order to improve the picking of skewed products. If additional DOF were introduced to the end effector in the horizontal direction (i.e., around the Y-axis), the tool-tip would be able, in some cases, to pick products without pushing. It would also be beneficial with an additional camera to capture depth images from above after the robot has picked a product to account for changes in the scene.
If not considering physical changes, another possibility is to calculate the new pose of the product after pushing. This could be achieved by reprogramming the code to update the collision map after the robot has reached the product.

Chapter 6

Conclusion

This thesis had the purpose of proposing a solution and looking into the challenges of picking products from distribution containers. Because of the wide scope of the problem, limitations and assumptions were set in order to evaluate a solution in a defined and delineate environment.

Three particular statistics stood out from the experiment. When the IoU_{PD} is low, e.g., $< 20\%$, the method cannot give a proper bounding box of the product. Without an accurate bounding box, the resulting cardboard evaluation is unreliable. The IoU_{PD} should, in this implementation, have a value above 80% to give a reliable product and cardboard detection.
The second observation is when the path planner fails to generate a robot-path in the simulator. In those cases in the experiment "Picking from Distribution Containers," the FP was $\approx 44\%$ compared to the successful ones with $\approx 1\%$. With an increased number of FP, it is more likely that the pick-point is occluded.
The last observation is with the use of Cartesian control. With the constraints and assumptions in this thesis, Cartesian control is sufficient and more reliable than RRT-Connect. The SR was 100% in simulation and 85% on the physical robot when using RRT-Connect, whereas the Cartesian control had 86.67% in both cases. The robustness of the Cartesian control, being able to pick the same products as the simulator, is a desired trait.

From the result of this thesis, it can be concluded that the proposed solution can work within the given assumptions and constraints. However, more work is necessary to see the performance of a larger number of products and product types. The products performing with an $\text{IoU}_{\text{PD}} > 80\%$ and Accuracy $> 90\%$, in both skewed and normal position, was "Oboy," "Granola," and "Juice."

Before continuing with the proposed solution, it is suggested to investigate possible machine learning methods, e.g., Mask R-CNN, as it could be superior in performance when detecting products and evaluating cardboard. For the robot path planner, a Cartesian control for products located in standard trays is from the conducted experiment considered to be "good enough." However, if introducing more complex shaped cardboard trays, the use of a path planner to find solutions could benefit, and the BIT* planner is an interesting planner to explore.

List of Figures

List of Tables

Appendix A

Experimental Products

All the products used in this thesis are listed here. All the product where used in the experiment "Picking from Distribution Containers" while A.1, A.2, A.3, A.4, and A.5 were used in the experiment "Picking Skewed Products."

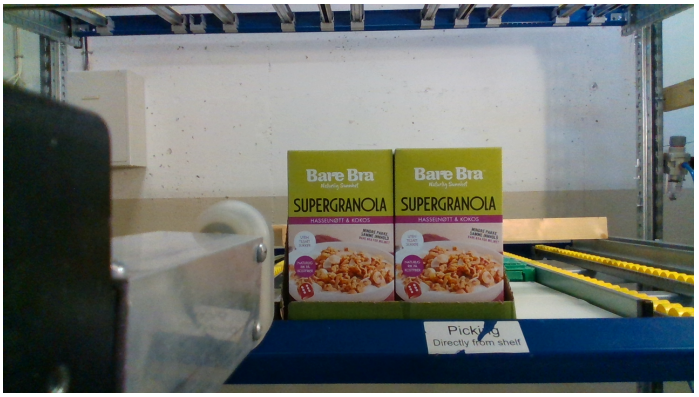
A.1 Oboy

(a) Image captured by the camera on robot.

(b) Reference image.

Figure A.1: Product Oboy: The figure display the Reference image and the captured image of the scene at the first iteration of the experiment.

A.2 Granola



(a) Image captured by the camera on robot.

(b) Reference image.

Figure A.2: Product Granola: The figure display the Reference image and the captured image of the scene at the first iteration of the experiment.

A.3 Juice

(a) Image captured by the camera on robot.

(b) Reference image.

Figure A.3: Product Juice: The figure display the Reference image and the captured image of the scene at the first iteration of the experiment.

A.4 Tea

(a) Image captured by the camera on robot.

(b) Reference image.

Figure A.4: Product Tea: The figure display the Reference image and the captured image of the scene at the first iteration of the experiment.

A.5 Blenda

(a) Image captured by the camera on robot.

(b) Reference image.

Figure A.5: Product Blenda: The figure display the Reference image and the captured image of the scene at the first iteration of the experiment.

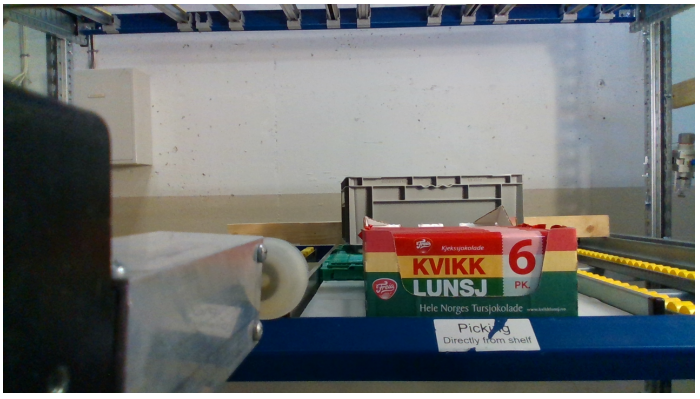
A.6 Asana

(a) Image captured by the camera on robot.

(b) Reference image.

Figure A.6: Product Asana: The figure display the Reference image and the captured image of the scene at the first iteration of the experiment.

A.7 Kvikklunch



(a) Image captured by the camera on robot.

(b) Reference image.

Figure A.7: Product Kvikklunch: The figure display the Reference image and the captured image of the scene at the first iteration of the experiment.

A.8 Sun

(a) Image captured by the camera on robot.

(b) Reference image.

Figure A.8: Product Sun: The figure display the Reference image and the captured image of the scene at the first iteration of the experiment.

Appendix B

Experimental Results

B.1 Experiment: Picking Products from Distribution Containers

Table B.1: Product: Oboy. Table describing the individual results from the experiment "Picking Products from Distribution Containers."

ID	Cat.	Pos.	TP	TN	FN	FP	Accuracy	IoU_{CD}	IoU_{PD}
oboy01	N/A	Row 1	25.75%	73.52%	0.63%	0.10%	99.27%	98.14%	92.84%
oboy02	N/A	Row 2	15.47%	74.43%	2.12%	7.99%	89.90%	74.27%	92.05%
oboy03	N/A	Row 1	23.67%	74.23%	1.97%	0.13%	97.90%	94.55%	92.30%
oboy04	N/A	Row 1	25.79%	74.15%	0.00%	0.05%	99.95%	99.86%	91.42%
oboy05	N/A	Row 2	15.25%	82.84%	1.91%	0.00%	98.09%	93.32%	93.72%
oboy06	N/A	Row 2	15.25%	81.36%	3.39%	0.00%	96.61%	88.91%	94.63%
oboy07	N/A	Row 3	9.43%	85.68%	2.01%	2.87%	95.11%	80.23%	83.73%
oboy08	N/A	Row 3	4.38%	87.63%	6.21%	1.77%	92.01%	63.53%	84.80%
oboy09	N/A	Row 3	4.90%	87.25%	5.70%	2.16%	92.15%	65.08%	94.72%
oboy10	N/A	Row 4	0.00%	95.13%	4.87%	0.00%	95.13%	47.56%	83.57%
oboy11	N/A	Row 4	0.00%	96.40%	3.60%	0.00%	96.40%	48.20%	86.32%
oboy12	N/A	Row 4	0.00%	96.19%	3.81%	0.00%	96.19%	48.09%	96.83%

Table B.2: Product: Tea. Table describing the individual results from the experiment "Picking Products from Distribution Containers."

ID	Cat.	Pos.	TP	TN	FN	FP	Accuracy	IoU_{CD}	IoU_{PD}
tea01	Plastic	Row 1	14.19%	78.20%	5.59%	2.01%	92.40%	78.13%	91.72%
tea02	Plastic	Row 1	15.32%	77.15%	4.15%	3.37%	92.48%	79.09%	90.56%
tea03	Plastic	Row 2	0.00%	93.10%	6.90%	0.00%	93.10%	46.55%	94.70%
tea04	Plastic	Row 2	3.39%	90.36%	4.99%	1.25%	93.75%	64.35%	90.34%
tea05	Plastic	Row 3	0.00%	13.78%	0.00%	86.22%	13.78%	6.89%	3.05%
tea06	Plastic	Row 3	0.00%	56.39%	0.00%	43.61%	56.39%	28.19%	4.95%
tea07	Plastic	Row 4	0.00%	74.96%	0.00%	25.04%	74.96%	37.48%	93.35%
tea08	Plastic	Row 4	2.84%	85.16%	2.34%	9.67%	88.00%	53.39%	86.57%
tea09	Plastic	Row 5	5.14%	79.09%	4.97%	10.80%	84.23%	53.97%	83.01%
tea10	Plastic	Row 5	0.00%	79.76%	0.00%	20.24%	79.76%	39.88%	96.99%

Appendix C

Illustration Images

C.1 Collision Point Cloud

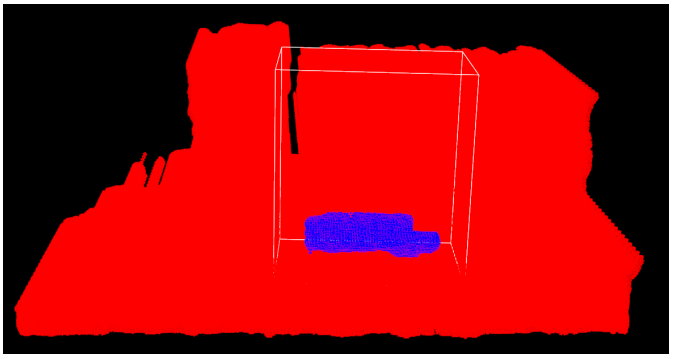


Figure C.1: The figure displays the *Collision Point Cloud* consisting of the different sub point clouds. The red area is the *Occlusion Point Cloud*, consisting of everything within the FOV. The white cube is the area removed by the *Product Point Cloud*, and the blue area is the *Cardboard Point Cloud* inserted due to the possibility of removed with the product.

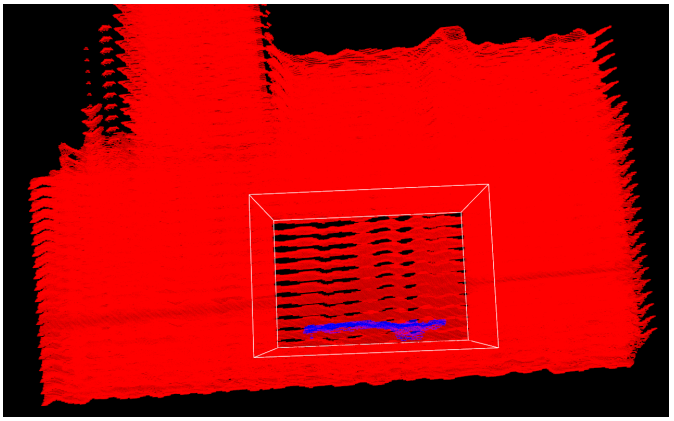


Figure C.2: The figure displays the same information as figure C.1, but the problem with the cardboard being removed is clearer, as it resides within the bounding box of the *Product Point Cloud*.

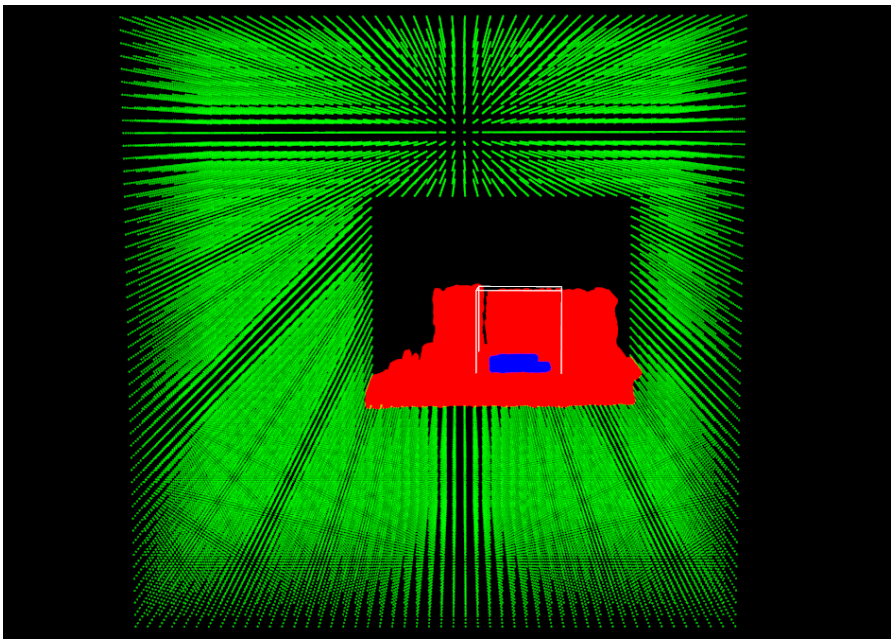


Figure C.3: The figure displays the *Restriction Point Cloud* (green points), restricting movement outside the shelf compartment. Inside the green qube, the red *Occlusion Point Cloud*, blue *Cardboard Point Cloud* and the white bounding box of the product is displayed.

C.2 Labelbox Images

(a) The output from Labelbox when labeling the product.

(b) The output from Labelbox when labeling the cardboard.

Figure C.4: The figure displays the output images from Labelbox. Figure (a) is the ground truth image for comparing the product detection performance, while (b) is the ground truth image used for comparing the cardboard detection performance.

C.3 Flow Chart of Image Processing

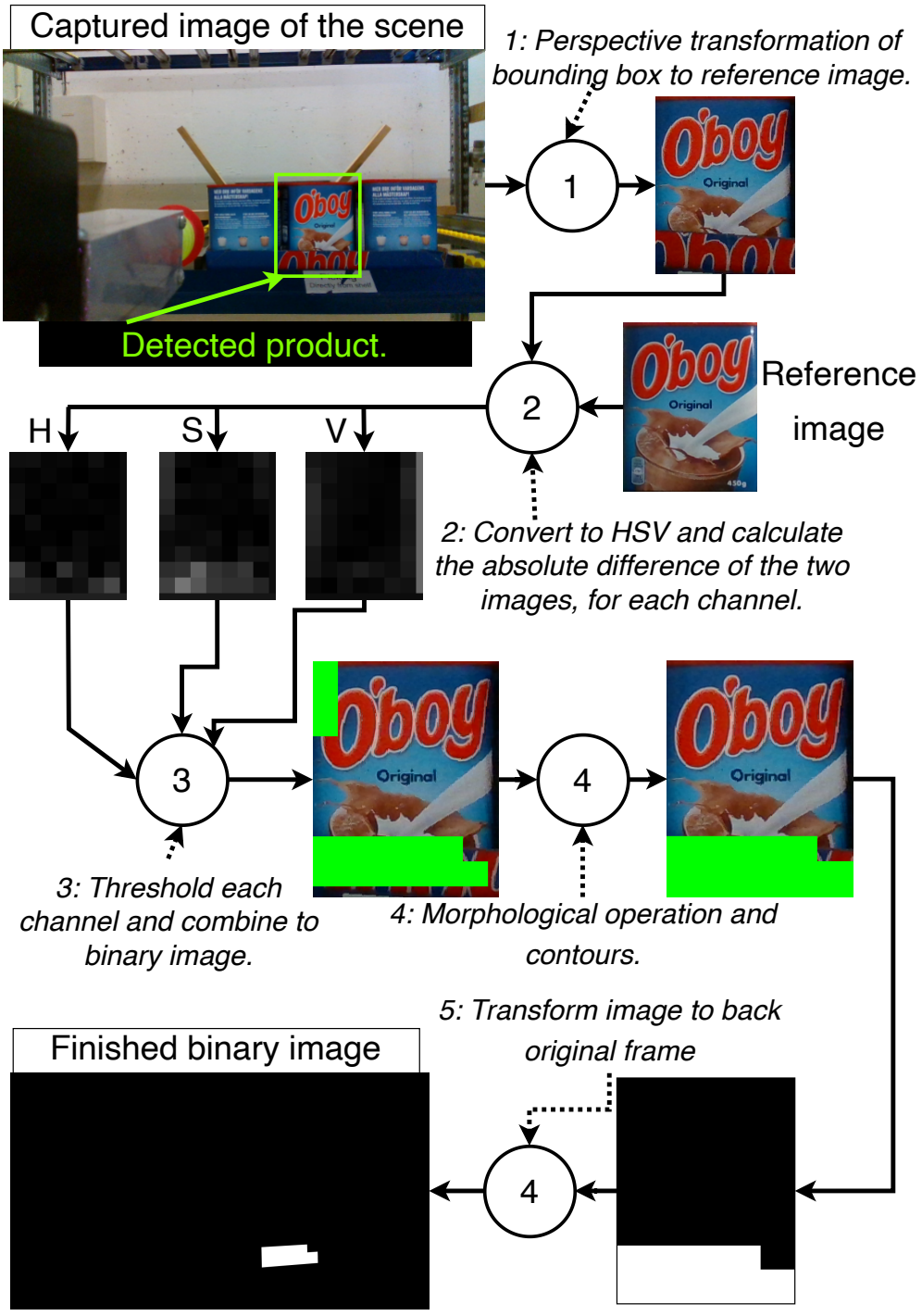
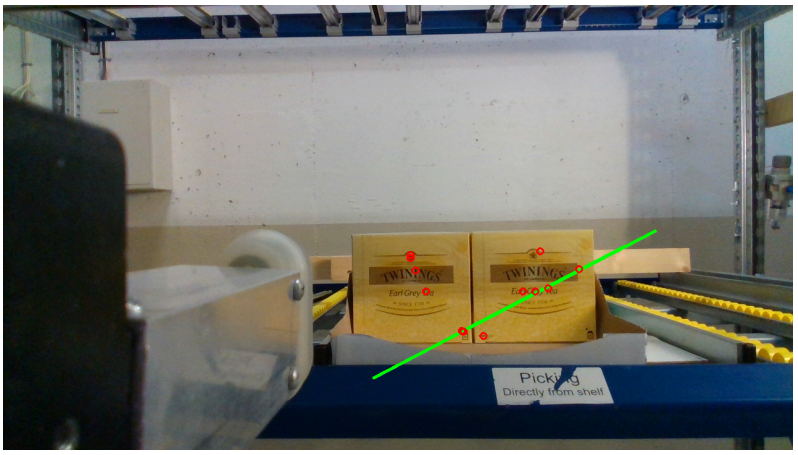


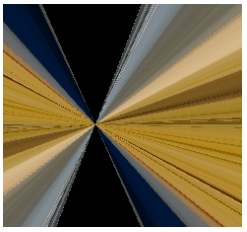
Figure C.5: The figure shows a principle flow chart for estimating the cardboard. ***1:*** The product was detected with SIFT, and the estimated region was transformed to the same perspective as the reference image. ***2:*** The reference image and the scene image were converted to the HSV color space, and the absolute difference calculated on each channel. ***3:*** For each channel, a threshold was set to determine if the regions differs enough to be considered cardboard. ***4:*** The binary image undergoes morphological operations, and the largest contour is kept. ***5:*** The binary image is transformed back to the original scene image.

C.4 Distortion

C.4.1 Product: tea05, Experiment: Picking from Distribution Containers



(a) The captured image with SIFT estimation of the product.



(b) The estimated product from figure (a).

Figure C.6: The figure display the SIFT estimation green line in (a), with the corresponding bounding box. The image from the bounding box is illustrated in (b), and yield a distorted image. This product got an IoU_{PD} of 3.05%.

C.4.2 Product: tea06, Experiment: Picking from Distribution Containers

(a) The captured image with SIFT estimation of the product.

(b) The estimated product from figure (a).

Figure C.7: The figure display the SIFT estimation green line in (a), with the corresponding bounding box. The image from the bounding box is illustrated in (b), and yield a distorted image. This product got an IoU_{PD} of 4.95%.

C.5 Cardboard Estimation Failure

C.5.1 Product: sun09, Experiment: Picking from Distribution Containers

(a) Reference image. (b) Scene image. (c) Estimated cardboard.

Figure C.8: The figure display the product sun09 with the reference image (a), the bounding box of the product used for cardboard detection in (b), and the estimated cardboard in (c). The product are from the experiment "Picking from Distribution Containers" and have an $\text{IoU}_{\text{PD}} = 92.5\%$ and FP $= 22.66\%$.

C.5.2 Product: asana03, Experiment: Picking from Distribution Containers

(a) Reference image. (b) Scene image. (c) Estimated cardboard.

Figure C.9: The figure display the product asana03 with the reference image (a), the bounding box of the product used for cardboard detection in (b), and the estimated cardboard in (c). The product are from the experiment "Picking from Distribution Containers" and have an $\text{IoU}_{\text{PD}} = 89.86\%$ and FP $= 15.76\%$.

C.6 Contours Failure

C.6.1 Product: oboy06, Experiment: Picking Skewed Products

(a) Estimated cardboard.

(b) Estimated cardboard after morphological operations.

(c) Resulting estimate after keeping largest contours.

Figure C.10: The figure illustrates the possible drawback of the cardboard estimation procedure. If the FP pixels get clustered in a continuous area and the number of FP > TP, the method will keep the FP and discard the TP, as seen in (c).

C.7 Cardboard Estimation

C.7.1 Product Type: Oboy, Experiment: Picking from Distribution Containers

(a) Oboy03, Row 1

(b) Oboy05, Row 2

(c) Oboy08, Row 3

(d) Oboy11, Row 4.

Figure C.11: The figures shows the cardboard's prominence for each row, highlighted in green. In row 1 (a), the cardboard is clearly visible, but for row 4 (d), this is not the case, and the cardboard estimation might not detect any cardboard.

Appendix D

Code

The program developed in this thesis is dependent on Pickr's confidential code. Because of this, the attached `main.cpp` is not compilable and is only attached to get some insight into the code. The illustration in figure 3.3 displays the simplified structure of the program.

D.1 main.cpp

The code in the `main.cpp` file is the main code of the system. The code is commented to some extent. This file uses some other classes created in this thesis is. Those classes are not attached since they are explained in the thesis and cannot compile due to dependencies. The mentioned classes are:

- `SiftCode`: The class for object detection with SIFT. As explained in section 3.3.2.
- `Hsv`: The class for estimating the cardboard. As explained in section 3.3.3.
- `OcclusionMap`: In short, generate the collision map. As explained in section 3.4.
- `ImageProsessing`: Helper methods to execute image processing functions. E.g., `convertBGRtoHSV()`.

In addition to the classes mentioned above, the `main.cpp` use one of the already existing files from Pickr. This class has not been created during this thesis but is essential for controlling the robot. The file is the `SystemInterface` class. The class is used for controlling and moving the physical robot.

Bibliography

[1] Link to video recording of experiment. URL https://www.dropbox.com/sh/4x3rbxs13knqqxc/AACQTE2wsxFphN00c5eUD_Uma?dl=0. Accessed on 2020-06-06.

[2] Pickr.ai. Pickr.ai: Web-page. URL https://www.pickr.ai. Accessed on 2020-04-25.

[3] Statistics Norway. SSB: Classification of Standard Industrial Classification, . URL https://www.ssb.no/en/klass/klassifikasjoner/6. Accessed on 2020-06-16.

[4] Statistics Norway. SSB Statistics, . URL https://www.ssb.no/en/statbank/sq/10036591. Accessed on 2020-04-25.

[5] Swisslog. Swisslog: A shopper ' s Guide to E-Grocery Fulfillment. URL https://www.swisslog.com/en-us/business-solutions/e-grocery.

[6] Brick Meets Click. How the online grocery market is shifting. 2018. URL www.brickmeetsclick.com.

[7] Fabric. All roads lead to online grocery: The state of online grocery in 2019 & critical challenges for the road ahead. 2020.

[8] Haifei Zhu, Yuan Yik Kok, Albert Causo, Keai Jiang Chee, Yuhua Zou, Sayyed Omar Kamal Al-Jufry, Conghui Liang, I. Ming Chen, Chien Chern Cheah, and Kin Huat Low. Strategy-based robotic item picking from shelves. *IEEE International Conference on Intelligent Robots and Systems*, 2016-Novem:2263–2270, 2016. ISSN 9781509037629. doi: 10.1109/IROS.2016.7759354.

[9] Quanquan Shao, Jie Hu, Weiming Wang, Yi Fang, Wenhai Liu, Jin Qi, and Jin Ma. Suction Grasp Region Prediction Using Self-supervised Learning for Object Picking in Dense Clutter. *2019 IEEE 5th International Conference on Mechatronics System and Robots, ICMSR 2019*, pages 7–12, 2019. ISSN 9781728122236. doi: 10.1109/ICMSR.2019.8835468.

[10] Hung Pham and Quang-Cuong Pham. Critically fast pick-and-place with suction cups. *2019 International Conference on Robotics and Automation (ICRA)*, pages

3045–3051, May 2019. doi: 10.1109/ICRA.2019.8794081. URL https://ieeexplore.ieee.org/document/8794081/. Accessed on 2020-05-05.

[11] Simon Marnburg Eriksen. Visual guided robotic picking system for the grocery industry. Master's thesis, University of Stavanger, Norway, 2017.

[12] Hussein Mnyusiwalla, Pavlos Triantafyllou, Panagiotis Sotiropoulos, Maximo A. Roa, Werner Friedl, Ashok M. Sundaram, Duncan Russell, and Graham Deacon. A Bin-Picking Benchmark for Systematic Evaluation of Robotic Pick-and-Place Systems. *IEEE Robotics and Automation Letters*, 5(2):1–1, 2020.

[13] Yu Sun, Joe Falco, Nadia Cheng, Hyouk Ryeol Choi, Erik D. Engeberg, Nancy Pollard, Maximo Roa, and Zeyang Xia. Robotic Grasping and Manipulation Competition: Task Pool. In Yu Sun and Joe Falco, editors, *Robotic Grasping and Manipulation*, volume 816, pages 1–18. Springer International Publishing, Cham, 2018. ISBN 978-3-319-94567-5 978-3-319-94568-2. doi: 10.1007/978-3-319-94568-2_1. URL http://link.springer.com/10.1007/978-3-319-94568-2_1. Accessed on 2020-05-05.

[14] Nikolaus Correll, Kostas E. Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M. Romano, and Peter R. Wurman. Analysis and observations from the first Amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2018. doi: 10.1109/TASE.2016.2600527.

[15] Andy Zeng, Shuran Song, Kuan Ting Yu, Elliott Donlon, Francois R. Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, Nima Fazeli, Ferran Alet, Nikhil Chavan Dafle, Rachel Holladay, Isabella Morena, Prem Qu Nair, Druck Green, Ian Taylor, Weber Liu, Thomas Funkhouser, and Alberto Rodriguez. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3750–3757, 2018. ISSN 9781538630815. doi: 10.1109/ICRA.2018.8461044.

[16] Michel Breyer, Fadri Furrer, Tonci Novkovic, Roland Siegwart, and Juan Nieto. Comparing task simplifications to learn closed-loop object picking using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 4(2):1549–1556, 2019. doi: 10.1109/LRA.2019.2896467.

[17] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning Synergies between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning. *IEEE International Conference on*

Intelligent Robots and Systems, pages 4238–4245, 2018. ISSN 9781538680940. doi: 10.1109/IROS.2018.8593986.

[18] Sandy H. Huang, Martina Zambelli, Jackie Kay, Murilo F. Martins, Yuval Tassa, Patrick M. Pilarski, and Raia Hadsell. Learning Gentle Object Manipulation with Curiosity-Driven Deep Reinforcement Learning. *arXiv:1903.08542 [cs]*, March 2019. URL http://arxiv.org/abs/1903.08542. Accessed on 2020-05-20.

[19] Miroslav Kubat. *An Introduction to Machine Learning*. Springer International Publishing, Cham, 2017. ISBN 978-3-319-63912-3 978-3-319-63913-0. doi: 10.1007/978-3-319-63913-0. URL http://link.springer.com/10.1007/978-3-319-63913-0. Accessed on 2020-06-16.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, June 2017. doi: 10.1109/TPAMI.2016.2577031.

[21] Wei Yi, Yaoran Sun, Tao Ding, and Sailing He. Detecting retail products in situ using CNN without human effort labeling. *arXiv preprint arXiv:1904.09781*, 2019.

[22] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *arXiv:1703.06870 [cs]*, January 2018. URL http://arxiv.org/abs/1703.06870. Accessed on 2020-05-07.

[23] Shengfan Wang, Xin Jiang, Jie Zhao, Xiaoman Wang, Weiguo Zhou, and Yunhui Liu. Vision Based Picking System for Automatic Express Package Dispatching. *arXiv:1902.08951 [cs]*, April 2019. URL http://arxiv.org/abs/1902.08951. Accessed on 2020-05-20.

[24] Kostas E. Bekris and Rahul Shome. Asymptotically Optimal Sampling-based Planners. *arXiv:1911.04044 [cs]*, 2020. URL http://arxiv.org/abs/1911.04044. Accessed on 2020-04-25.

[25] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, 1:473–479, 1999. doi: 10.1109/ROBOT.1999.770022. URL http://ieeexplore.ieee.org/document/770022/. Accessed on 2020-06-02.

[26] Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[27] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Batch Informed Trees (BIT*): Sampling-based Optimal Planning via the Heuristically

Guided Search of Implicit Random Geometric Graphs. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074, May 2015. doi: 10.1109/ICRA.2015.7139620. URL http://arxiv.org/abs/1405.5848. Accessed on 2020-06-01.

[28] J.J. Kuffner and S.M. LaValle. RRT-connect: An efficient approach to single-query path planning. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, 2:995–1001, 2000. doi: 10.1109/ROBOT.2000.844730. URL http://ieeexplore.ieee.org/document/844730/. Accessed on 2020-05-03.

[29] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24:381–395, 1981. doi: 10.1145/358669.358692.

[30] Richard Szeliski. *Computer Vision*. Texts in Computer Science. Springer London, London, 2011. ISBN 978-1-84882-934-3 978-1-84882-935-0. doi: 10.1007/978-1-84882-935-0. URL http://link.springer.com/10.1007/978-1-84882-935-0. Accessed on 2020-06-02.

[31] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, second edition, 2004. ISBN 978-0-511-18618-9. URL http://public.eblib.com/choice/publicfullrecord.aspx?p=256634. Accessed on 2020-06-04.

[32] Dongqing Li, editor. *Encyclopedia of Microfluidics and Nanofluidics*. Springer New York, New York, NY, 2015. ISBN 978-1-4614-5488-5 978-1-4614-5491-5. doi: 10.1007/978-1-4614-5491-5. URL http://link.springer.com/10.1007/978-1-4614-5491-5. Accessed on 2020-06-16.

[33] Shamik Sural, Gang Qian, and Sakti Pramanik. Segmentation and histogram generation using the HSV color space for image retrieval. *IEEE International Conference on Image Processing*, 2:589–592, 2002. ISSN 0780376226. doi: 10.1109/icip.2002.1040019.

[34] Linda Shapiro and George Stockman. *Computer Vision*. 2000.

[35] Gerhard X Ritter and Joseph N Wilson. *Computer Vision Algorithms in Image Algebra*. CRC Press, second edition, 2000. ISBN 0-8493-0075-4.

[36] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999. doi: 10.1109/iccv.1999.790410.

[37] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 91–110, 2004.

[38] Tony Lindeberg. Scale Invariant Feature Transform. *Scholarpedia*, 7(5):10491–10491, 2012. doi: 10.4249/scholarpedia.10491.

[39] Marius Muja and David G. Lowe. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, November 2014. ISSN 0162-8828, 2160-9292. doi: 10.1109/TPAMI.2014.2321376. URL http://ieeexplore.ieee.org/document/6809191/. Accessed on 2020-06-04.

[40] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, 2006. ISBN 978-0-511-54687-7 978-0-521-86205-9. doi: 10.1017/CBO9780511546877. URL https://www.cambridge.org/core/product/identifier/9780511546877/type/book.

[41] Sertac Karaman and Emilio Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *arXiv:1105.1186 [cs]*, May 2011. URL http://arxiv.org/abs/1105.1186. Accessed on 2020-05-09.

[42] Li Huang and Becker T. Aaron. Rapidly Exploring Random Tree (RRT) and RRT*, 2018. URL http://demonstrations.wolfram.com/RapidlyExploringRandomTreeRRTAndRRT/.

[43] Jesper Dall and Michael Christensen. Random Geometric Graphs. *Physical Review E*, 66(1):016121, July 2002. ISSN 1063-651X, 1095-3787. doi: 10.1103/PhysRevE.66.016121. URL http://arxiv.org/abs/cond-mat/0203026. Accessed on 2020-06-05.

[44] Jason Jason Lu and Minlu Zhang. Heuristic search. In Werner Dubitzky, Olaf Wolkenhauer, Kwang-Hyun Cho, and Hiroki Yokota, editors, *Encyclopedia of Systems Biology*, pages 885–886. Springer New York, New York, NY, 2013. ISBN 978-1-4419-9863-7. URL 10.1007/978-1-4419-9863-7_875.

[45] BIT* Video: YouTube. URL https://www.youtube.com/watch?v=TQIoCC48gp4. Accessed on 2020-06-04.

[46] Jonathan D. Gammell, Timothy D. Barfoot, and Siddhartha S. Srinivasa. Batch Informed Trees (BIT*): Informed Asymptotically Optimal Anytime Search. *The International Journal of Robotics Research*, 39(5):543–567, April 2020. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364919890396. URL http://arxiv.org/abs/1707.01888. Accessed on 2020-06-04.

[47] Labelbox. Labelbox: Web-page. URL https://labelbox.com. Accessed on 2020-05-21.

[48] ROS. ROS: Web-page, . URL https://www.ros.org. Accessed on 2020-05-31.

[49] ROS. About ROS, . URL https://www.ros.org/about-ros/. Accessed on 2020-06-18.

[50] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: An open-source Robot Operating System. *ICRA workshop on open source software*, 3:5, 2009.

[51] ROS. ROS: Master, . URL http://wiki.ros.org/Master. Accessed on 2020-06-13.

[52] ROS. ROS: Node, . URL http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes. Accessed on 2020-06-13.

[53] ROS. ROS: Topic, . URL http://wiki.ros.org/Topics. Accessed on 2020-06-13.

[54] Intel. Intel RealSense D435. URL https://www.intelrealsense.com/depth-camera-d435/. Accessed on 2020-06-28.

[55] Intel. Intel® RealSense™ Camera D400 series Product Family: Datasheet, 2019.

[56] Intel. Intel ® RealSense ™ Camera: Depth Testing Methodology, 2018.

[57] Sai Sriparasa. *JavaScript and JSON Essentials*. Packt Publishing, Limited. ISBN 978-1-78328-604-1.

[58] RapisJSON. RapisJSON. URL https://rapidjson.org. Accessed on 2020-04-25.

[59] Point Cloud Library. PCL: Web-page, . URL http://pointclouds.org. Accessed on 2020-04-25.

[60] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1–4, 2011. ISSN 9781612843865. doi: 10.1109/ICRA.2011.5980567.

[61] Gary Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. URL https://opencv.org. Accessed on 2020-04-25.

[62] OpenCV. OpenCV License, . URL https://opencv.org/license/. Accessed on 2020-04-25.

[63] MoveIt! MoveIt!: Web-page. URL https://moveit.ros.org. Accessed on 2020-04-25.

[64] OMPL. Open Motion Planning Library: Web-page, . URL https://ompl.kavrakilab.org. Accessed on 2020-05-02.

[65] OMPL. OMPL: License, . URL https://ompl.kavrakilab.org/license.html#bsdlicense. Accessed on 2020-04-24.

[66] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013. doi: 10.1007/s10514-012-9321-0.

[67] OcotMap. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. URL https://octomap.github.io. Accessed on 2020-05-27.

[68] ROS. RViz, . URL http://wiki.ros.org/rviz. Accessed on 2020-06-28.

[69] OpenCV. OpenCV Tutorial: Features2D + Homography, . URL https://docs.opencv.org/3.4/d7/dff/tutorial_feature_homography.html. Accessed on 2020-06-01.

[70] Point Cloud Library. PCL Tutorial: Plane model segmentation, . URL https://pcl-tutorials.readthedocs.io/en/latest/planar_segmentation.html. Accessed on 2020-06-01.

[71] Gilbert Strang. *RES.18-001 Calculus Online Textbook*. Massachusetts Institute of Technology: MIT OpenCourseWare, 2005. URL https://ocw.mit.edu/resources/res-18-001-calculus-online-textbook-spring-2005/. Accessed on 2020-06-01.

[72] Jack B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press, Princeton, N.J, 2002. ISBN 978-0-691-10298-6.

[73] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. *arXiv:1902.09630 [cs]*, April 2019. URL http://arxiv.org/abs/1902.09630. Accessed on 2020-05-23.

[74] Harald Thirud Skutvik. Evaluating Computer Vision Methods for Detection and Pose Estimation of Textureless Objects. Master's thesis, University of Stavanger, Norway, 2019.

[75] Maria Petrou and Costas Petrou. *Image Processing: The Fundamentals*. Second edition, 2010. ISBN 978-0-470-74586-1.