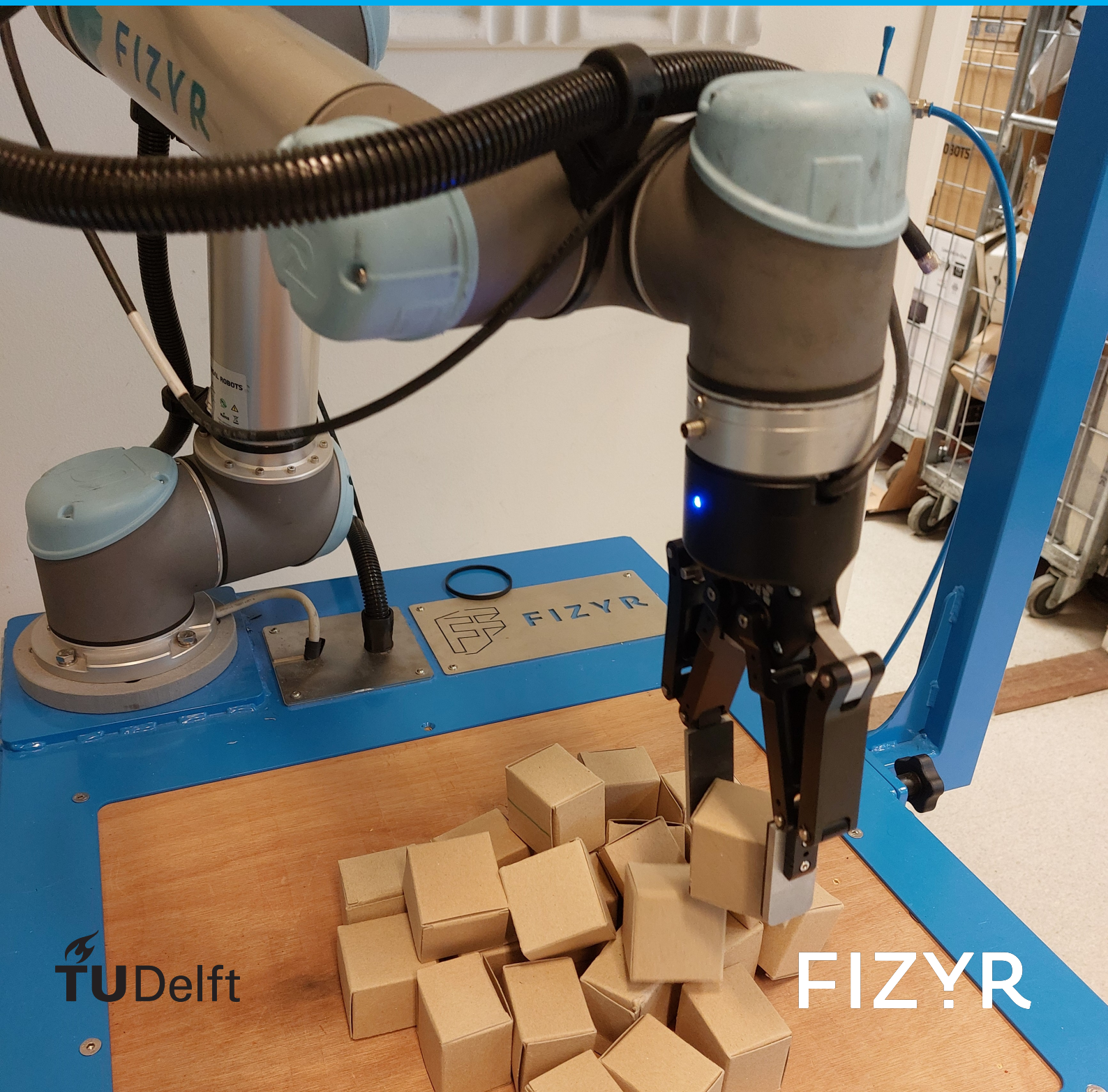
Master of Science Thesis

Grasp-RCNN for a two-fingered pinch-gripper

A multiple RCNN approach

L.R. Lipman



TUDelft

FIZYR

This page is intentionally left blank.

Grasp-RCNN for a two-fingered pinch-gripper

A multiple RCNN approach

by

L.R. Lipman

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday, Augustus 29, 2022 at 14:00.

In collaboration with Fizyr B.V.

Student number:	42868683	
Course Code:	ME-51035	
Project duration:	November 1, 2020 – Augustus 29, 2022	
Thesis committee:	Prof. dr. W. (Wei) Pan,	TU Delft, supervisor
	Dr. ir. G. (Gerwin) Smit,	TU Delft
	Ir. Nuno Mota,	Fizyr, supervisor

TU Delft

FIZYR

This page is intentionally left blank.

Abstract

Introduction - Grasping unknown objects is an important ability for robots in logistic environments. While humans have an excellent understanding of how to grasp objects because of their visual perception and understanding of the 3D world, robotic grasping is still a challenge. Due to the fast-growing development of deep learning methods, it is now possible to train deep neural networks on this grasp task.
Objective - This thesis proposes a bin-picking pipeline that uses deep learning to take care of the perception and estimation task. The pipeline can predict grasps for known and unknown objects with a two-fingered pinch-gripper in real-world environments in a single object and multi-object scenes.
Method - A grasp annotation tool has been developed to generate a wide variety of grasps in the training data that are antipodal and collision-free. Together with annotated objects, the generated grasps are used to train Grasp-RCNN. The developed Grasp-RCNN combines an object- and a grasp-detection network to predict objects masks and grasps, and a decision algorithm that picks the best-estimated grasp based on a grasp score.
Results - Robotic experiments demonstrate that the proposed method allows a robot gripper to grasp both known and unknown objects in single-object and multi-object scenes with a total success-rate of 89.7% and 81.0% with average process-times of 616 ms and 739 ms per scene respectively. In a bin-picking scene a success-rate of 87.5% with a process-time of 1235 ms is achieved.
Conclusion - These results indicate that the proposed Grasp-RCNN is able to grasp known and unknown objects with an accuracy that is comparable to the state-of-the-art. For production purposes, the speed of the network still can be improved.

This page is intentionally left blank.

Preface

It is hard to make the boat go as fast as you want to. The enemy of course, is resistance of the water, as you have to displace the amount of water equal to the weight of the men and equipment, but that very water is what supports you and that very enemy is your friend. So is life: the very problems you must overcome also support you and make you stronger in overcoming them.

George Yeoman Pocock

The problems and frustrations I had during this project I've overcome, which resulted in a better thesis. The thesis in front of you marks both the end of the research I conducted to finish the Mechanical Engineering Master at the TU Delft and the fun student time I've had in Delft. It summarizes the work I've done in the period between November 2020 and August 2022. This is a bit longer than I expected due to the persuasion of a dream that I still have: getting into the National Rowing team. To accomplish both goals, I had to combine this project with elite-level rowing, which was due to the 24-hour training schedule, rowing tests and races all around the world, not always a perfect fit and quite challenging.

With this project, I combined my interests in robotics and machine learning and I'm thankful that Fizyr gave me the opportunity to work on this grasp problem. I would like to thank both Vivek and Nuno for their supervision during the project. It was very interesting to learn from your experiences and use that in this project.

I also would like to thank Dr. Wei Pan and Dr. Ir. Gerwin Smit for being my TU Delft supervisor. Wei for allowing me to pursue a project that I thought was very interesting while giving me the freedom and trust to organize my project and Gerwin for the useful tips and tricks to make this report even better.

During the project, I could count on the endless support of my friends, the Pierres, and the Spechtjes. A special mention for Menno for all the time we spent on and off the water with rowing, coffee breaks, and study sessions. At last a special thanks to Babette, my mom, and my sister for the patience, endless love, and support during this thesis project. You all kept me motivated!

L.R. Lipman
Delft, Augustus 2022

This page is intentionally left blank.

Contents

1 Introduction

In the past few years, e-commerce has grown fast and after the corona lockdown, the e-commerce became even more significant. Warehouses and logistics with order pickers had to shift towards automation and robotic solutions to keep up with the growth. The products in e-commerce have a big diversity in goods, sizes, shapes and materials, and therefore a general approach is needed for pick-and-place tasks. Humans are good at these tasks, because of their visual perception and understanding of the 3D world. However, for robots that rely solely on RGB- and depth images, understanding the environment and the 3D world is complex. For the pick-and-place task, the robot needs to understand the distinction between objects and their orientation to grasp them. In order to automate this task, robots need to outperform human order-pickers on performance, speed, and accuracy.

With the introduction of deep learning in computer vision, robots are getting more acquainted with the handling and understanding of visual information. It is now possible to train robots on these grasp tasks with high picking accuracies. A general grasp system consists of a combination of object detection, grasp estimations, grasp planning and a control system for the robot. While the planning and the control of an end-effector are as relevant for a correct grasp as grasp detection, this thesis focuses on grasp detection solely. The planning and control systems used in this thesis are therefore mentioned, but not discussed.

1.1. Background

To pick up objects, the robot and end-effector are vital components. In industry, a wide variety of robotic arms are developed with a 5-7 Degrees of Freedom (DoF) movement to ensure a high order of flexibility in position and orientation of the end-effector. End-effectors with one or multiple suction cups are used to grasp objects using a pressure difference [19], while two- or three-fingered pinch-grippers grasp with force- or shape-enclosure [18, 35]. Unlike suction-cups, pinch-grippers are able to grasp objects that are permeable to air such as stuffed animals, and objects with a rough surface. Research is also done on dexterous grippers [28, 39] to grasp objects with pneumatic or active changing surfaces in hand-shaped grippers.

To guide an end-effector to a grasp-position, a 6-DoF gripper pose is essential to grasp a target object. A 6-DoF grasp pose consists of three position and three orientation variables. The pose is dependent on the location of the grasped object, but could be limited by the location of other objects. When the gripper direction is perpendicular to the table, the grasp pose could be simplified to a 4-DoF planar grasp pose with three positions and one orientation variable. If the height is also fixed or 2D grasps are predicted, the planar grasp can be described by 3-DoF.

The information that is required to generate grasp predictions is obtained from both RGB images and pointcloud data. Based on the RGB images a distinction can be made between objects on a pixel level, while the pointcloud data contains the information of the pixels in Cartesian space. In

pick-and-place tasks, depth information is a crucial element to understand the shape, size and orientation of the object. For 6-DoF grasps this information is used to create valid poses with different orientations. For planar grasps, only the orientation around the z-axis is predicted. Depth information is then used to obtain information on the approach height.

For actual grasp predictions, researchers have developed different methods to predict grasp poses. For known objects, it is possible to grasp objects when they are in a predefined location, or when they pass a sensor. It is also possible to store 3D information of objects with predefined grasp locations in a database and match the pointcloud and RGB-image with methods such as shape-completions [17, 30] or 6-DoF object-pose estimation [36]. These methods match templates or pointcloud features to extract the predefined grasp estimation from existing databases. When the amount of stored objects increases, this shape matching can be time-consuming, which could make the process slower. Some research is done on transferring grasps from known objects to unknown objects with similar geometry [27, 20]. However, when the geometry of the unknown object is not comparable with a known object, this will not work. With the growing number of goods in warehouses, it is desired that these novel objects can be grasped without the use of stored grasp information in a database.

To account for unknown objects, grasps could be estimated with an analytical method [31] that computes grasps with available information of an RGB-image and depth information and achieves grasp accuracies of 88.2% and 77.0% for single and multi-object environments respectively. Research is also done on the use of neural networks for grasp predictions. For planar grasps, [13] was the first to apply deep-learning methods on grasp predictions with a two-step classifications-based cascade system. Their work was quickly improved by [22] in both accuracy and precision by using spatial locations instead of a sliding window approach. [11] proposed a single-step predicting network where they use ResNet [7] instead of AlexNet [10] to decrease the run-time of grasp prediction. Furthermore, they assume only one object with a single predicted grasp and improved the accuracy significantly. [6] combined both visual and tactile sensing to their model to enhance the robot's perception and grasping skills, achieving 93.2% and 89.1% on single and multi objects respectively on objects from the Cornell Database.
The above-mentioned methods are achieving up to 93.2% accuracy on grasp predictions on the Cornell dataset but lack results in real-life experiments which is done by the following methods: [37] introduced a RoI-GD, a two-step approach, to detect grasps based on Region of Interest (RoI). Within these RoI' grasps are predicted. For cluttered scenes, they combine this with the Faster-RCNN network to classify objects, such they can decide which objects will be grasped and achieve an accuracy of 92.5 and 83.8 in real-world single and multi objects scenarios. [40] introduced the orientated anchor boxes, which allow the use of only one anchor box with different orientations. The orientation, position and size of the anchor box are predicted with respect to their original orientation. [4] used the work of [40] by extending the multi-grasp predictor with a direct dependency between the score evaluations and the regressed predictions. With this method, an accuracy of 92.4% is achieved in single-object environments. In contrast with the previous methods, [15] proposed a grasp detection based on key-points instead of anchor-based or rectangular-based grasps achieving 94% and 87% in single and multi-object scenes and is the current state-of-the-art in 2D-planar grasping.

These neural networks needs to be trained on datasets that contain annotated grasps. For 2D planar grasps, only a few datasets exist. Some of them were created by manual annotations [25], while [14, 21] went a step further and annotated grasps by robotic grasp attempts. [21] created a dataset of 50k data points with a Baxter robot within 700 hours, while [14] used up to 14 real robot arms to collect 800k data points within two months. In both cases, the authors successfully trained a CNN network to detect grasp locations from their data. To get rid of the time-consuming robotic experiments, [18] created a synthetic dataset of over 6.7M images by using annotated data from simulations. However, for every image in the aforementioned datasets, only one grasp-annotation per image is given, which is sub-optimal for grasp predictions in cluttered scenes. The Cornell Grasp Dataset [9] and the Jacquard dataset [5] contain multiple grasps per image. The Cornell Grasping Dataset contains 885 RGB-D images with 8019 hand-annotated grasp rectangles. To increase the number of images, and thus grasp data, [5] created the Jacquard dataset with 54k images and 1.1 million grasps based on an automatized algorithm. Due to the large variety of objects in the Jacquard dataset, this should generalize to a variety

of objects. However, this dataset is only available for academic purposes and could therefore not be used in this thesis.

1.2. Research Objective

As discussed, few researchers actually put their network for planar grasps in cluttered scenes into practice and the grasp databases that are used contain one grasp per object, are small, or are not publicly available. Therefore the objective of this thesis is a bin-picking pipeline that uses deep learning to generate a complete solution for bin-picking with a two-fingered pinch-gripper in real-world environments. The pipeline must be able to grasp known and unknown objects in a single-object and multi-object environment. The gripper can not collide with the object, surrounding objects or the environment.
To achieve this goal a grasp generation tool needs to be developed that is able to generate grasps that are feasible to grasp and collision free. Second a neural network needs to be developed that predicts feasible grasps. At last, from the predicted grasps, a ”best-grasp” needs to be selected which is grasped.

1.3. Outline

This report sequentially addresses the above-mentioned objectives. Chapter 2 explains the grasp annotation tool. A complete overview of the network’s architecture, the training preparation and the post-processing is given in chapter 3. Chapter 4 explains the experimental setup and the results and also includes the observations. In chapter 5 the results are discussed and recommendations for future work are made. This thesis work is concluded in chapter 6 and the extra material is added to the appendix.

This page is intentionally left blank.

2

Data Generation

A neural network is trained to make predictions, based on training data. In this research the Grasp-RCNN will be making grasp predictions around objects, so the training data needs to contain locations of grasps and objects. The process of adding information to images is called annotation. The dataset from Fizyr used in this thesis already contains the object annotation. This chapter focuses on the generation of the grasp annotations. First, the different data annotation types are discussed, followed by the requirements and inputs of the grasp annotation tool. At last the working of the grasp annotation tool is explained.

2.1. Data Annotation Types

The data annotation of grasps can be done in three different ways: physical annotation, computer simulation and manual annotation. The explanation of these three methods will follow in this section.
Physical annotation is done using a physical robot setup that attempts grasps at random locations in the scene. The advantage of this method is that the correct obtained grasps annotations are as accurate as they can be. A disadvantage of this method is that it requires a lot of time and resources. In literature, [21] created 50k grasp annotations with robotic attempts in 700 hours, while [14] used fourteen robot arms to collect 800k annotations in two months.
To overcome this time issue, computer simulations can also be used to generate images and their corresponding annotations. The benefit of this method is that the exact location in the simulated environment is known and that a lot of different scenarios can be created in a relatively short amount of time. [18] used this method to create a synthetic dataset of 6.7M images and 6.7M data points, containing one grasp per image. [5] also used synthetic data to create the Jacquard grasp dataset with 54k images and 1.1 million grasps based on their automatizing algorithm. However, learning from synthetic data could lead to undesired performance due to the gap between synthetic and real image distributions [1].
The last type of annotation is manual annotation. With this method, a human operator annotates the grasps in the images by hand. This method is faster than generating data using physical annotations but is slower than computer-simulated annotations. While it is slightly less accurate than a computer simulation, the annotations are done on real data which should give better results.

In this thesis, manual annotation is combined with computer simulation by automatizing an analytical algorithm to create grasp annotations. This has as advantage that a lot of grasps can be generated in a relatively short amount of time on real image data. Also, the synthetic gap is not present, since real images are used. A disadvantage of this method is that the grasps are dependent on the quality of the analytical method. The requirements of the analytical model and working of the analytical model are discussed in the following sections.

2.2. Grasp Requirements

The analytical method needs to generate feasible grasp annotations based on the annotated objects and the pointcloud. An object annotation consists of a list of pixel coordinates (x, y) that describes the

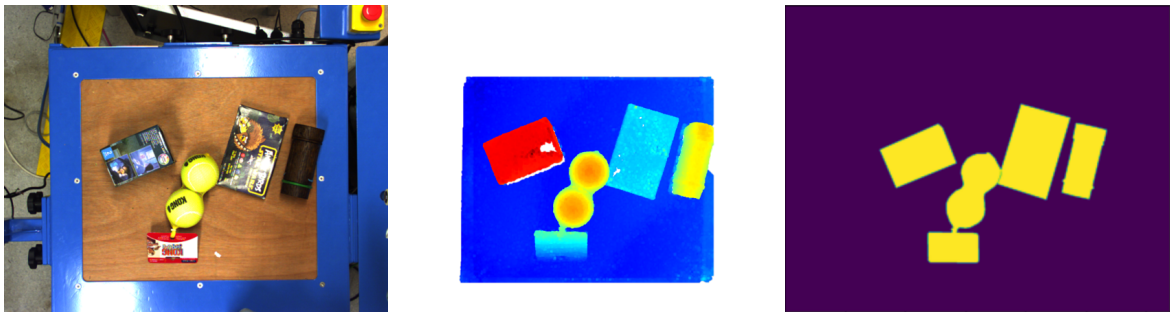
contour of an object in the image. An image could contain more than one object. From the object annotation, the grasp annotations are created. To acquire the required data, the following assumptions/rules should be implemented in the grasp annotation tool:

- Images could contain zero, one, or multiple objects.
- Objects could be partly outside the grasp area; grasp should be generated in the graspable area.
- Objects could be in contact or overlap with other objects.
- The gripper can not collide with the object they grasp, surrounding objects or the environment.
- The grasp must be antipodal around the grasped object.

A grasp annotation tool with the required functionalities is created in Python. Further details about the implementation of requirements is given in the following section.

2.3. Input Data

The input of the grasp annotation tool is an RGB image and its corresponding pointcloud and object annotations. An object annotation consists of a list with the x and y coordinates describing the contour of the object. In the tool, also an object mask is used. An object mask is a binary image with the same size as the original RGB image with ones at the pixels representing the object, and zeros on the pixels that are not. An overview of the input of the algorithm can be seen in fig. 2.1. The pointcloud does not cover the full image, but only the graspable area on the table. Every point of the pointcloud corresponds to one pixel of the RGB image.



(a) (b) (c)

Figure 2.1. The input of the network (from left to right) is (a) an RGB image and its (b) corresponding pointcloud and (c) the total set of object masks. The pointcloud does not cover the whole RGB image, but only the area on the wooden table. The colors in the pointcloud indicate how high the object is.

2.4. Grasp Annotations

In this section, the grasp annotation is discussed. First, the grasp rectangle is explained, followed by the explanation of three different methods that are used to create the 2D grasps based on the input described in section 2.3. The grasps are generated based on a Principle Component Analysis, the skeleton method [31], and the Center of Mass (CoM) of the objects. Initially, the grasps have a length that is equal to the maximum distance between the fingers in pixels. After the grasps are generated, the distance between the fingers is trimmed and the grasps that collide or are not antipodal (see section 2.5.3, are removed. These steps are explained in section 2.5.

2.4.1. Grasp Representation

For the grasp annotation, an orientated grasp rectangle first described by [9] is used. This grasp rectangle takes into account the location, the orientation and the dimensions of the gripper. The gripper representation is described as:

$$g = c_x, c_y, w, l_f, \theta_{grasp} \tag{2.1}$$

with (c_x, c_y) as gripper center, w as width of the fingers, l_f as length between fingers and θ_{grasp} as grasp orientation. The gripper rectangle is visualized in fig. 2.2.

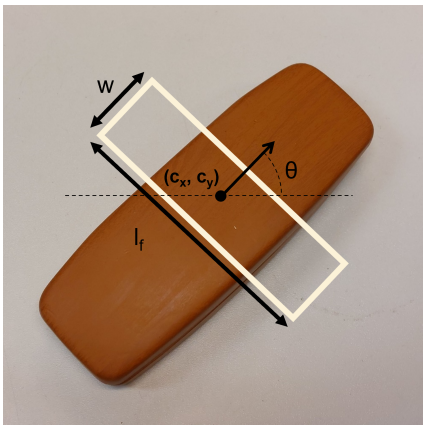


Figure 2.2. A grasp is represented by a rectangle with a center (c_x, c_y), dimensions w and l_f, and orientation θ_{grasp}

In the following sections, this grasp rectangle is used to create various grasps around objects.

2.4.2. Principle Component Analysis

The first method to generate grasps is based on a Principle Component Analysis (PCA) of the object masks.

"The principal components of a collection of points in coordinate space are a sequence of p unit vectors, where the i^{th} vector is the direction of a line that best fits the data while being orthogonal to the first $i-1$ vectors" - [33].

For a collection of 2D points, e.g. for the binary masks of objects, this represents a set of two 2D-orthogonal vectors called the major and minor axis. The major axis represents the direction of the mask with the most variance, e.g. for box-shaped objects the major axes should be parallel to the longest side. The grasps are then generated orthogonal to the major and minor axes with interval steps of 25 pixels. In fig. 2.4a, grasps based on this method are shown. When shapes become more complex, it is possible that the principal component analysis is not generating antipodal grasps. For these kinds of shapes, the skeleton method is used.

2.4.3. Objects Skeleton

To have grasps cover the whole object mask, thus also around protrusions, the object skeleton method can be used. [38] described a method that creates a skeleton of a mask. By making several passes around the border of the mask and removing pixels on the border of the mask, one can create a one-pixel wide, fully connected, representation of the mask. This skeleton has as property that the shortest distance to the contour of the mask is equal on both sides of the line. A visual representation of a skeleton is shown in fig. 2.3. The mask of the horse in the image is reduced to a set of lines that maintain connectivity. Along the skeleton, grasps are created to cover the object.

The grasp rectangles will be generated orthogonal to the orientation of the skeleton. To calculate the orientation of the skeleton at the center of the grasp rectangle the following calculation is used: let $\mathbf{q}_n$ be the set of pixels that represent the skeleton. Take a circular region with radius r around every n^{th} point of $\mathbf{q}_n$. Let $\mathbf{p}_m$ represent the pixels of the skeleton, inside this circular region. The orientation of the skeleton at the grasp center $\mathbf{q}_n$ is then calculated by:

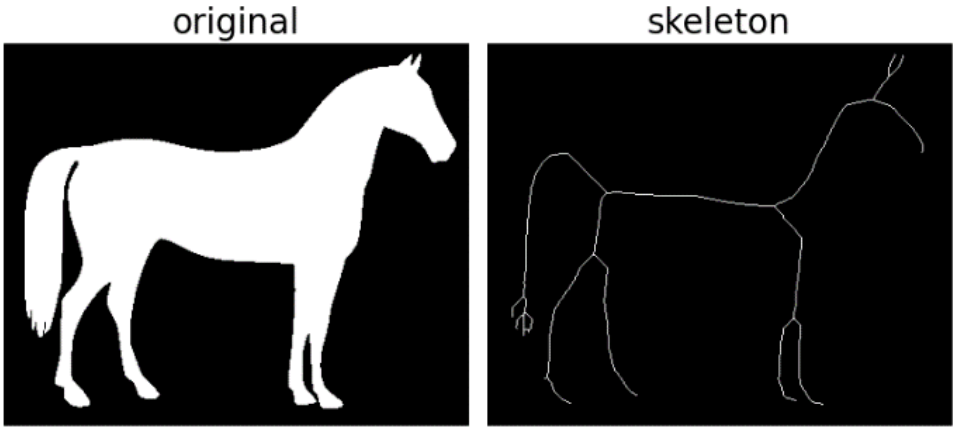


Figure 2.3. A visualization of a mask's skeleton. The algorithm makes successive passes along the border of the mask, removing pixels with the condition that the connection of the object stays intact. (Image from [26])

$$\mu_x = \frac{\sum_{i=1}^{m} \mathbf{p}_i(x)}{m} \quad (2.2)$$

$$\mu_y = \frac{\sum_{i=1}^{m} \mathbf{p}_i(y)}{m} \quad (2.3)$$

$$\sigma_x = \sum_{i=1}^{m} \left(\mu_x - \mathbf{p}_i(x)\right)^2 \quad (2.4)$$

$$\sigma_y = \sum_{i=1}^{m} \left(\mu_y - \mathbf{p}_i(y)\right)^2 \quad (2.5)$$

$$\sigma_{xy} = \sum_{i=1}^{m} \left(\mu_x - \mathbf{p}_i(x)\right)\left(\mu_y - \mathbf{p}_i(y)\right) \quad (2.6)$$

$$b = \frac{\sigma_x - \sigma_y + \sqrt{\left(\sigma_x - \sigma_y\right)^2 + 4\sigma_{xy}^2}}{2\sigma_{xy}} \quad (2.7)$$

$$\theta_{skel} = \tan^{-1}(b) \quad (2.8)$$

The grasp rectangle has center $\mathbf{q}_{n:x,y}$ and a slope orthogonal to θ_{skel}. In fig. 2.4b the grasps based on this method are shown.

2.4.4. Rotation around CoM

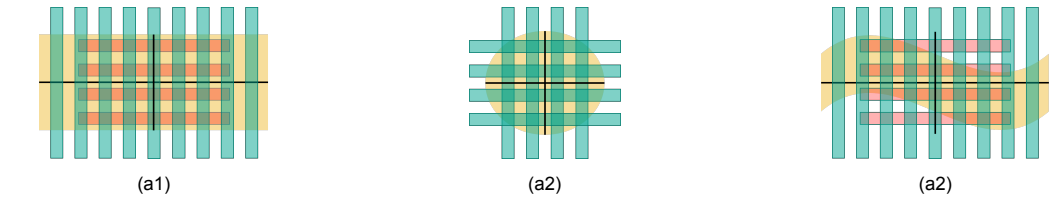
For round objects, the methods described above do not create a good grasp. For a good grasp, the fingers must be antipodal to the object (see section 2.5.3 for more information). When the PCA method is used on round objects, it creates only antipodal grasps around the center of the mask. When the skeleton method is used for a perfectly round object, the skeleton consists of only one point and a short line for an ellipse.

To create feasible grasps for round objects, grasps are created around the CoM. The CoM of a binary mask is calculated by taking the average x and y coordinates of all the pixels with value 1. During the grasp generation, the grasp rectangle is rotated around the CoM of the mask with stepsize θ_{rot_step}:

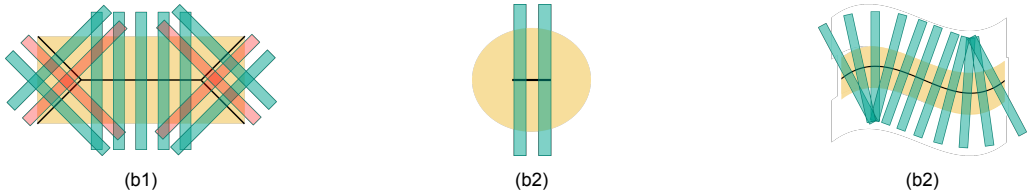
$$\theta_{rot} = \{\theta_{rot\_step} k \mid k \in [0..180]\} \quad (2.9)$$

This method creates feasible, antipodal grasps for round objects, in fig. 2.4c the grasps based on a rotation around the CoM are shown.

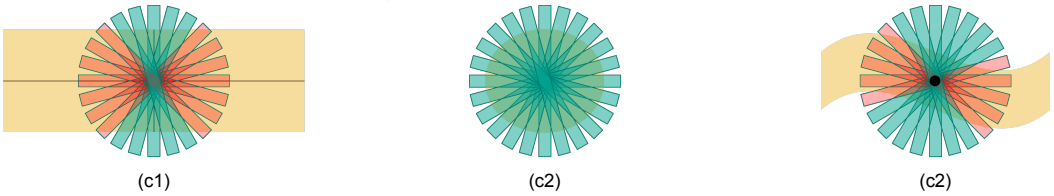
The grasps generated by the above-mentioned methods are all generated with a maximum distance between the fingers. Before the next step, the grasps where at least one of the ends of the grasp rectangle collide with the object, are removed, This step is visualized in fig. 2.4 by the red rectangles.



(a) Grasps created using the PCA-method, the major and minor axis are indicated in black.



(b) Grasps created using the skeleton-method. The skeleton is indicated by the black lines



(c) Grasps created around the CoM of objects. The CoM is indicated with a black dot.

Figure 2.4. Grasp rectangles (green) are created around an object (yellow) based on three different methods, from top to bottom row: (a) the PCA-method, (b) the object skeleton and (c) the CoM. Objects have, from left to right, (1) a rectangular shape, (2) a round shape and (3) a complex shape. Red rectangles indicate grasps that are colliding with the object and are therefore removed before the next step.

2.5. Grasp Annotation Optimization

The grasps annotations generated with one of the before-mentioned methods are based on, but not optimized for the shape of individual masks. The generated grasps are also not 'aware' of the surrounding objects. Therefore the generated grasps will go through four steps to filter, remove and optimize the generated grasps.

2.5.1. Grasp Trimming

The first step is to trim the predicted grasps such that the fingers come closer to the object. This step is performed to decrease the chance that the fingers are colliding with other objects. The decreasing of the finger distance is done at both ends individually as shown in fig. 2.5. As result, the finger distance l_f and the gripper center might alter.

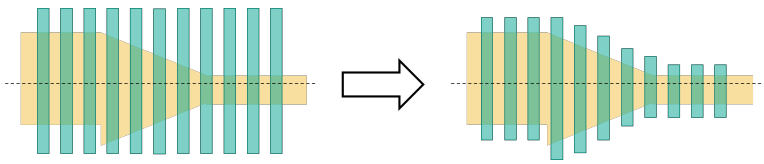


Figure 2.5. Each end of the grasp rectangle is trimmed to get a smaller grasp around the mask. For simplicity only grasps generated with the PCA-method are shown.

2.5.2. Collision Detection

After the grasps are trimmed, the algorithm checks if the grasps rectangles collide with other objects. A grasp is considered in collision with another object if at least one end of the grasp rectangle, thus the location of the gripper finger, overlaps the mask of another object. In fig. 2.6 two objects are close together. The grasp rectangles that overlap the mask of the other object are indicated as red and removed.

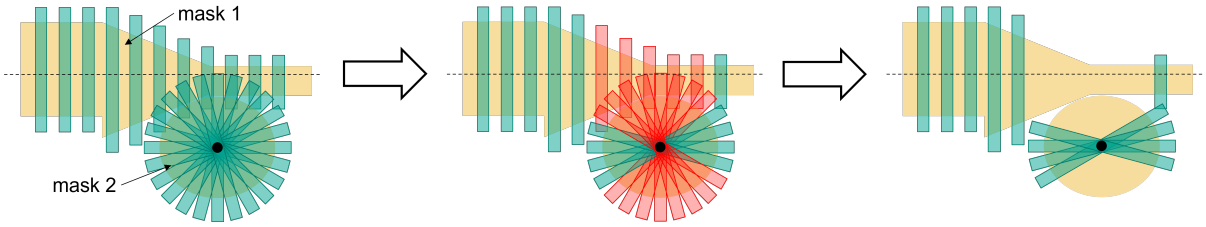


Figure 2.6. (l) Two object masks are close together (m) the rectangles of which one of the ends overlaps the other mask are indicated in red and (r) removed.

2.5.3. Antipodal Grasps

A two-fingered pinch-gripper can grasp an object if the contact points are antipodal. Antipodal points have surface normal vectors which are co-linear and in opposite directions. If such a pair exist, a force-closed grasp is guaranteed [3].

To calculate if a grasp meets this antipodal requirement a few calculations are made. The first step is to calculate the contact points between the gripper and the object. Second, the surface normal vectors at these locations are calculated. The last step is to compare the orientation of these normal vectors with the orientation of the grasp. If this difference exceeds a certain limit, the grasps are discarded. These steps are explained in more detail below.

The first step is to determine the contact points of the grasp on the object. Let each grasp g_i be presented by a grasp-line lg with center (c_x, c_y), length l_f and orientation θ_{grasp}. Each object mask $\mathbf{O}_n$ consists of a set of annotated boundary points $\mathbf{B}_i$, which contains the contour coordinates. The intersections between grasp-line lg and the contour of $\mathbf{O}$ are defined as contact-points $\mathbf{cp}_1$ and $\mathbf{cp}_2$. A visualization of this is shown in fig. 2.7. Note that the number of contact points must be even. An odd number indicates that one of the fingers is on top of the object mask. These grasps are not used and thus discarded. With more than four contact points, only the outer $\mathbf{cp}_i$'s are used.

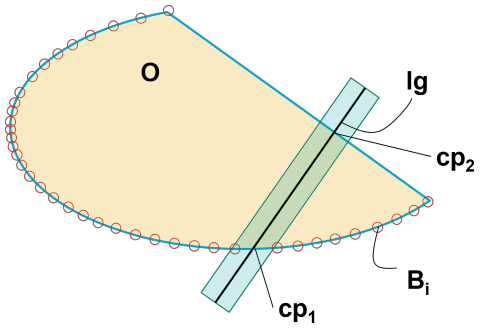


Figure 2.7. An object mask $\mathbf{O}$ (yellow) with a contour (blue) defined by boundary points $\mathbf{B}_i$ (red). The grasp rectangle $\mathbf{g}$ (green), is presented by grasp-line lg. lg intersects the contour of $\mathbf{O}$ at $\mathbf{cp}_1$ and $\mathbf{cp}_2$.

The second step is to calculate the normal-vectors at contact-points $\mathbf{cp}_1$ and $\mathbf{cp}_2$. This calculation is given for the normal-vector at $\mathbf{cp}_1$, but is similar for $\mathbf{cp}_2$. A visualization is given in fig. 2.8. Define $\mathbf{BC}_n$ as a subset of $\mathbf{B}$ that lie within a circular region C with radius $r_{antipodal}$ around center $\mathbf{cp}_1$. Let $\mathbf{B}_{1,j}$ and $\mathbf{B}_{1,j-1}$ be the boundary points from $\mathbf{B}$ that are on each side of $\mathbf{cp}_1$. The tangent-vector at the $\mathbf{cp}_1$ can be calculated by re-using eqs. (2.2) to (2.8). In these equations, $\mathbf{p}_n$ represents the pixels of which the orientation is calculated, and is dependent on the size of subset $\mathbf{BC}_N$, see eq. (2.10). If the size of $\mathbf{BC}_n$ is less than two, $\mathbf{p}_n$ is defined as set $[\mathbf{B}_j, \mathbf{B}_{j-1}]$, otherwise as $\mathbf{BC}_n$. Orthogonal to the tangent vector the normal vector is calculated.

$$\mathbf{p}_{n(x,y)} = \begin{cases} \mathbf{BC}_n & \text{if } |\mathbf{BC}_n| \geq 2 \\ \mathbf{B}_{j-1}, \mathbf{B}_j & \text{if } |\mathbf{BC}_n| < 2 \end{cases} \tag{2.10}$$

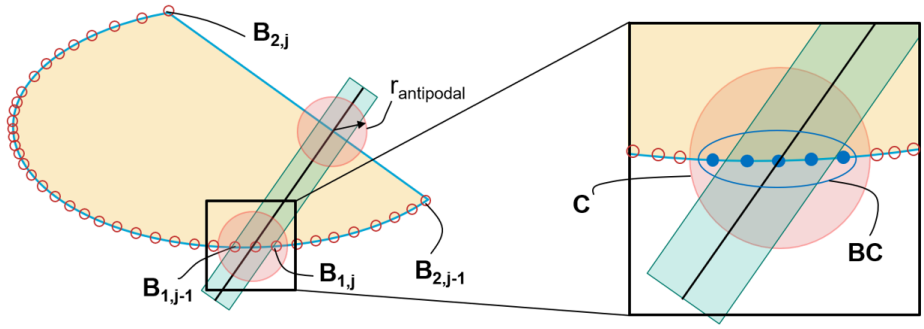


Figure 2.8. At $\mathbf{cp}_1$ and $\mathbf{cp}_2$ a circular area C with radius $r_{antipodal}$ is defined in red. On each side of $\mathbf{cp}_i$, $[\mathbf{B}_{1,j}, \mathbf{B}_{1,j-1}]$ and $[\mathbf{B}_{2,j}, \mathbf{B}_{2,j-1}]$ are indicated. At $\mathbf{cp}_1$ multiple boundary points are located inside C indicated with blue dots. For the calculation of the surface normal at $\mathbf{cp}_1$, $\mathbf{BC}$ is used. At $\mathbf{cp}_2$, there are no boundary points inside the circle, $|\mathbf{BC}| = 0$, so the surface normal is calculated using $\mathbf{B}_{2,j}$ and $\mathbf{B}_{2,j-1}$.

The last step is to compare the orientations of the surface-normal and grasp-line lg. In fig. 2.9, a visual representation is given. With the assumption that the contact at $\mathbf{cp}_i$ is a point-contact with friction, force closure is achieved if and only if lg lies within a friction cones θ_{cone} at both $\mathbf{cp}_1$ and $\mathbf{cp}_2$. The size of θ_{cone} is dependent on the friction between the gripper and the object. According to the Robotiq manual [24], the silicon fingertips have a static friction coefficient on lubricated steel of 0.3. Since the static friction coefficient of lubricated steel is lower than the friction coefficient between silicon and materials such as plastic, wood, or cardboard, a friction coefficient of 0.3 is considered safe. Using eq. (2.11), a θ_{cone} of $15\,\mathrm{deg}$ will be used.

$$\theta_{cone} = \arctan(\mu) \tag{2.11}$$

$$\mathbf{g}_{antipodal} = \begin{cases} \text{True} & \text{if } |\theta_{cp_i} - \theta_{grasp}| < \theta_{cone} \\ \text{False} & \text{if } |\theta_{cp_i} - \theta_{grasp}| > \theta_{cone} \end{cases} \tag{2.12}$$

Only grasps where lg lies within the friction-cone at both $\mathbf{cp}_i$ are considered antipodal. Looking at fig. 2.9 at $\mathbf{cp}_2$, lg lies within the friction cone and is thus antipodal. At $\mathbf{cp}_1$, lg lies not inside the friction cone, so this contact is not antipodal. This grasp is thus removed from the grasp-set.

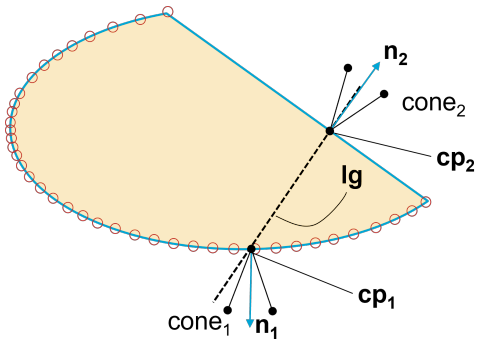


Figure 2.9. At each contact-point $\mathbf{cp}_i$, the surface-normal $\mathbf{n}_i$ (blue arrow) is calculated. Around each $\mathbf{n}_i$, a friction cone is drawn. A grasp is antipodal if at all $\mathbf{cp}_i$, lg lies within the friction cone. At $\mathbf{cp}_2$, lg lies within the friction cone, but at $\mathbf{cp}_1$ it is not. This grasp is thus not antipodal and will be removed from the generated data.

In fig. 2.10 the step for the antipodal check is visualized with the scenario that is used before. The middle part of mask 1 is inclined, so the normal-vectors are now under an angle. The surface normal vector and the friction cone are also shown. The grasp-line lg that does not lie within the friction cones, and are removed.

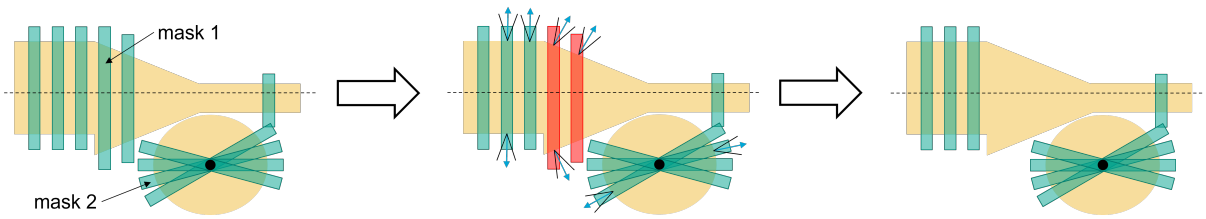
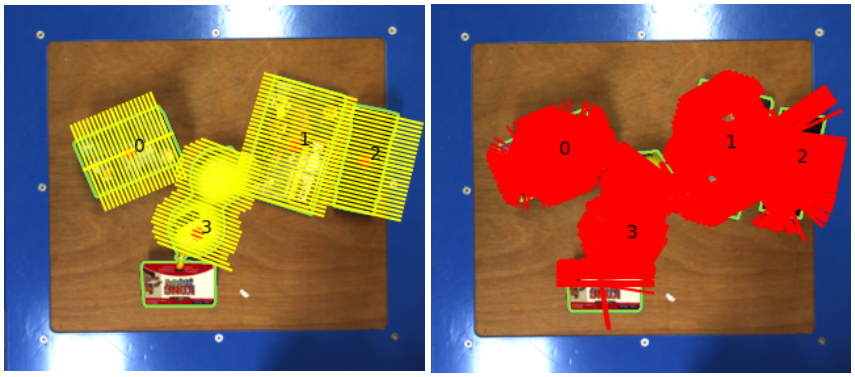


Figure 2.10. The grasps and masks from the collision check are passed through to the antipodal check (left). For each grasp, the surface normal vector and friction cone are calculated (only a few of them are shown), and grasps that do not lie within the friction cone are indicated in red and removed. (middle). A grasp annotation that is both collision-free and antipodal is passed through to the network (right).

This chapter gave an overview of the different steps in grasp generation. Figure 2.11 shows the different steps on a real image. In figs. 2.11a to 2.11c the generated grasps with the methods from section 2.4 are shown. On these generated grasps a collision and antipodal check is done. Figure 2.11d gives a visual summary of these steps. Both the white and green lines are according to the algorithm collision-free. It can be observed that a lot of grasps from figs. 2.11a to 2.11c are removed around the two right objects. The objects on the right are very close together and the grasp rectangle has overlap and is thus not considered collision-free. After the collision check, the antipodal check is performed. All the green lines in fig. 2.11d represent the collision-free and antipodal grasps. Grasps that were generated near the corners of boxes are removed. Only the green grasps are saved to train the network. The network architecture and the training will be discussed in chapter 3.



(a) Grasps generated by the PCA-method

(b) Grasps generated by the skeleton-method

(c) Grasps generated by the rotation method

(d) All these grasps are collision-free, the green lines indicate antipodal grasps

Figure 2.11. The grasps generated by the annotation tool on actual data are based on (a) the PCA method, (b) the skeleton method, and (c) the rotation method. After the grasp generation, the grasps are checked for collision with surrounding objects. It is also checked if the grasps are antipodal. Figure 2.11d shows all collision-free grasps in white and green. Grasps that are antipodal are indicated in green and are saved to train the network.

3

Grasp-RCNN

In this chapter, the neural network model will be discussed. First, a detailed overview of the architecture of the network is given, followed by the underlying parameters. At last, the post-process is discussed.

3.1. Architecture

The Grasp-RCNN consists of two multi-stage networks that take a single RGB image as input and produces grasps and object masks as output. A Mask-RCNN network [8] is used for the prediction of the masks, and a Faster-RCNN network [23] is used for the predictions of the grasps around objects. The RGB images are fed through several convolutional layers creating a Convolutional Neural Network (CNN). The CNN layers are pre-trained on the Microsoft COCO dataset [16] and use ResNet50 [7] as a backbone. The ResNet50 backbone creates different feature maps that are passed through to the Region Proposal Network (RPN) which produces a Region of Interest (RoI) with a score. A summary of the working of a CNN and RPN is given in appendix A. Based on the best scoring RoI, the information from the feature maps is processed to the further layers of the network. For every RoI, one particular prediction is made in the image. Since each object can have multiple grasp predictions, the RPN of the object detection and grasp detection had to be separated. Figure 3.1 shows an overview of the Grasp-RCNN architecture.

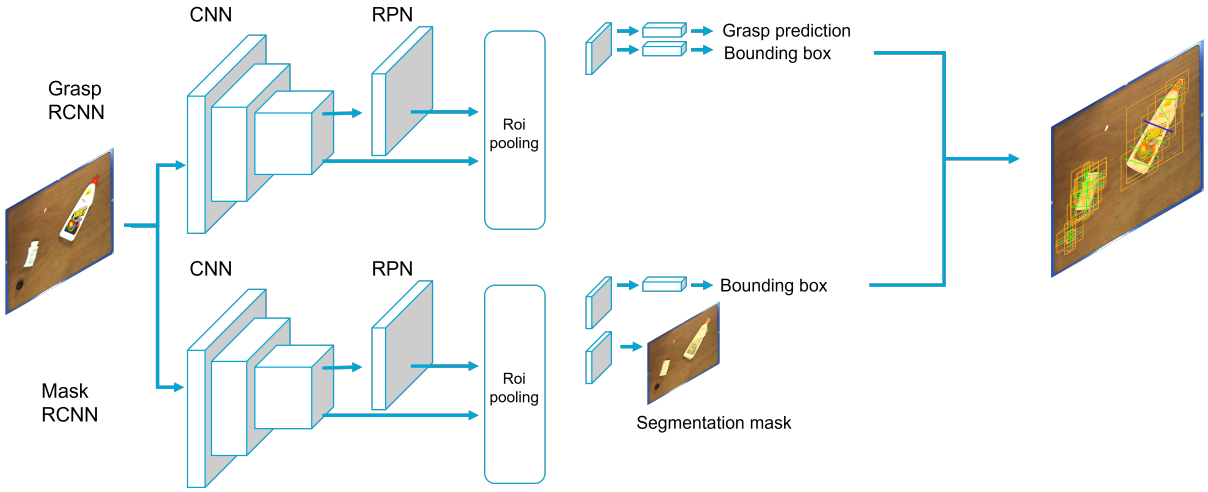


Figure 3.1. The Grasp-RCNN architecture is visualized. The architecture consists of a Faster-RCNN (top) and a MaskRCNN network (bottom) that consists of a pretrained network. Since multiple grasps are predicted for one object, the RPN could not be combined. The left side creates the different RoI proposals while the right side gives the output of a grasp and the segmentation.

3.2. Output Branches

The right side of the Grasp-RCNN consists of several output branches, each with its type of output. For each RoI the branches produce the following output:

- Faster-RCNN (Grasp detection)
  - Bounding box ($bbox$): A bounding box is a rectangle that enclosed the RoI from the RPN. The output is a list of $n \times 4$ values with the coordinates and the dimensions of the bounding box.
  - Grasp: For every bounding box, a grasp is predicted by the network. Since we make use of the RoI, the output of the grasp-prediction head is with respect to its surrounding bounding box. The output consists of $n \times 5$ values with the gripper representation $g = c_x, c_y, l_f, \theta_x, \theta_y$. The center coordinates are with respect to the top-right corner of the bounding box, the length as a percentage of the diagonal length of the bounding box, and θ_x and θ_y representing an x and y vector on the unit circle.
- Mask-RCNN (Object detection)
  - Bounding box ($bbox$): A bounding box is a rectangle that enclosed the RoI from the RPN. The output is a list of $n \times 4$ values with the coordinates and the dimensions of the bounding box.
  - Segmentation mask: For each bounding box a segmentation mask is predicted for the object that the bounding box is related to. The segmentation consists of ones for pixels that are predicted to be part of the object in the bounding box and zeros for the pixels that are not.

3.3. Training Preparation

The generated grasps from chapter 2 need to be transformed into data that the network understands. The Faster-RCNN and Mask-RCNN networks are based upon a Region Proposal Network (RPN) that looks at small patches of the images. In these patches, based on feature maps, a prediction is made of possible grasps/objects. During training, the RPN predicts in each patch a grasp representation and its bounding box. In this model, the bounding box is represented by:

$$bbox = [x, y, w, h] \tag{3.1}$$

with x and y as coordinates, and w and h as bounding box dimensions.

The size of the feature map is always the same, regardless of the size of the predicted bounding box. Therefore, the gripper representation that the network predicts on the feature map needs to be invariant to the size of the bounding box. The representation is made invariant based on the bounding box dimensions $[bbox_w, bbox_h]$. The orientation of the grasp is split into an x and y component. t_{grasp} is introduced as the ground-truth grasp.

$$bbox_{diag} = \sqrt{bbox_w^2 + bbox_h^2} \tag{3.2}$$

$$t_{cx} = \frac{c_x - bl_x}{bbox_w} \tag{3.3}$$

$$t_{cy} = \frac{c_y - bl_y}{bbox_h} \tag{3.4}$$

$$t_l = \frac{l}{bbox_{diag}} \tag{3.5}$$

$$t_{\theta_x} = \cos\left(\theta_{grasp}\right) \tag{3.6}$$

$$t_{\theta_y} = \sin\left(\theta_{grasp}\right) \tag{3.7}$$

$$t_{grasp} = t_{cy}, t_{cy}, t_{l_f}, t_{\theta_x}, t_{\theta_y} \tag{3.8}$$

3.4. Training Parameters

The Grasp-RCNN is trained on a single Nvidia Titan X GPU. As a training dataset, 765 images, 3883 objects, and 102275 grasps are used. The network is trained with a batch size of 4 and the learning rate set to 0.0001. The implementation is done in PyTorch 1.9.0. with TorchVision 0.10.0a0, CUDA 11.4 and cuDNN 8202. Both networks use ResNet50 as a backbone.

For the experiments, both networks are trained separate for 75 epochs with 1000 steps and an Adam optimizer with B_1 = 0.9, B_2 = 0.999 and $\epsilon = 1e-8$.

3.5. Loss Functions

The network learns from its training data by minimizing a loss function. Since a Mask-RCNN network and Faster-RCNN networks are trained separately, each network will have its own loss function. Since the networks are trained for multiple tasks the multi-task loss function is calculated by the sum of losses produces by each branch of the network. A single pass through the network produces multiple RoI, so the loss for each output branch is the sum of losses within each RoI. The different predicted RoI's are matched and compared against the closest ground-truth bounding box. An RoI is considered valid if the Intersection of Union (IoU) with the ground-truth bounding box is at least 0.5. If multiple RoI are matched with the same ground truth, only the RoI with the highest IoU score is used.

For the classification branch, the classification loss L_{cls} is defined for two classes (object *vs.* no object). In the work of [8, 23], this cross-entropy loss is defined as:

$$L_{cls} = -\sum_{i=1}^{n} y_f \log\left(p\left(y_i\right)\right) \tag{3.9}$$

For the bounding box and grasp predictions, the regression loss function is the same. As input for the bounding box regression, the following parameterization is used:

$$t_{bbox} = [t_x, t_y, t_w, t_h] \tag{3.10}$$

$$v_{bbox} = [v_x, v_y, v_w, v_h] \tag{3.11}$$

with t_{bbox} as ground-truth bounding box, v_{bbox} predicted bounding box and with x and y as the center coordinates and w, h as the bounding box dimensions. The ground-truth grasp is defined in eq. (3.8) where the predicted grasp is defined as v_{grasp}/

$$L_{bboxx} = \sum_{i \in \{x,y,\omega,h\}} L_{1;\,\text{smooth}}\left(t_{bbox\_i} - v_{bbox\_i}\right) \tag{3.12}$$

$$L_{grasp} = \sum_{i \in \{c_y,c_y,l_f,\theta_x,\theta_y\}} L_{1;\,\text{smooth}}\left(t_{grasp\_i} - v_{grasp\_i}\right) \tag{3.13}$$

$$L_{1;\,\text{smooth}}\left(x\right) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{3.14}$$

The mask branch generates a mask of dimensions N x N for each RoI and k class. The mask loss L_{mask} is defined as the average binary cross-entropy loss

$$L_{\text{mask}} = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log\left(p\left(y_i\right)\right) + \left(1 - y_i\right) \cdot \log\left(1 - p\left(y_i\right)\right) \tag{3.15}$$

with m as the number of pixels in the mask, y_i the ground-truth for pixel i and $p(y_i)$ as the probability of a pixel to be part of the mask.

The loss functions for the object detection and grasp detection networks are

$$L_{tot\_object} = \sum_{i=1}^{R_{object}} L_{cls}^{i} + L_{bbx}^{i} + L_{mask}^{i} \tag{3.16}$$

$$L_{tot\_grasp} = \sum_{i=1}^{R_{grasp}} L_{cls}^{i} + L_{bbx}^{i} + L_{grasp}^{i} \tag{3.17}$$

with $R_{objects}$ and R_{grasp} as the amount of valid RoI in the different networks.

3.6. Post Processing

To go from the predicted 2D-planer grasp to actual grasping a few post-processing steps are made. In the following sections, the post-processing of the predicted grasps is explained.

3.6.1. Object - Grasp matching

The output of the network is a list of n object masks and m grasp locations in pixels. Together with the pointcloud data, these are used to get real grasp locations. The first step is to match the predicted grasps with the predicted object masks. Every grasp of which a center lies on top of a mask is assigned to that particular mask. Grasps that do not have a corresponding mask, are removed. The next step is to calculate the grasp score of each mask, this will be discussed in the next section.

3.6.2. Grasp Score

To determine which grasp is the best, a grasp-score is assigned to each grasp. The grasp-score s_{grasp} is calculated based a distance-score s_{dist} and an angle-score s_{angle}. The distance-score is based on the distance between the gripper center and the CoM of the object mask. For the distance-score, a maximum distance d_{max} between the gripper center and the mask CoM of 300px is chosen since the biggest objects in the image had a length of around 600px. For the calculation of the angle-score, the smallest angle between the orientation of the gripper and the orientation of one of the principle axis of the object is taken. The total grasp-score is the average of the distance- and angle-score. The s_{grasp}, s_{dist} and s_{angle} are calculated with:

$$d_{CoM} = \sqrt{(gripper_y - CoM_y)^2 + (gripper_x - CoM_x)^2} \quad [px] \tag{3.18}$$

$$s_{dist} = \frac{d_{max} - d_{CoM}}{d_{max}} \quad [0\ldots1] \tag{3.19}$$

$$\alpha_{diff} = (mask_\theta - grasp_\theta)\ \%\ 180 \quad [degrees] \tag{3.20}$$

$$\alpha_{gap} = \alpha_{diff}\ \%\ 45 \quad [degrees] \tag{3.21}$$

$$s_{angle} = \frac{45 - \alpha_{gap}}{45} \quad [0\ldots1] \tag{3.22}$$

$$s_{grasp} = \frac{s_\alpha + s_d}{2} \quad [0\ldots1] \tag{3.23}$$

3.6.3. Sorting

With a list of grasps and their grasp score, the best grasp is picked. To reduce the change in a possible collision the highest objects should be picked first. Therefore the grasps are sorted first on the height of their corresponding object, and second on the individual grasp score of these grasps. The gripper will thus grasp the best grasp for the highest object.

3.6.4. Collision Check

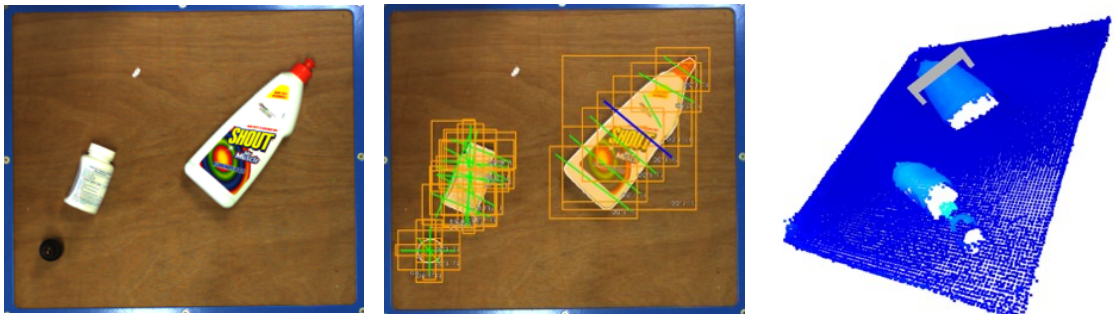
During the creation of the training data, a collision check is performed to create grasps that do not overlap with other object masks (section 2.5.2). Before the actual grasp, a second collision check is performed. The method checks the height data at the area covered by the left and right finger locations and compares it with the height data from the object. When this difference exceeds 0.01 m, a grasp is performed. Since the network relies on the depth data from the sensor, it is also checked if there is at least 50% data present in the area covered by the fingers. If this is not the case due to possible occlusions or bad sensor quality, the predicted grasp is removed.

3.6.5. Pixels to Grasp Coordinates

Until this point, everything is calculated in pixel coordinates. The pointcloud has a data point for every pixel in the RGB image above the table. The x and y coordinates at both ends of the blue line are used to calculate the gripper length l_f. The coordinates of the center of the blue line are used for the grasp location. The rotation of the gripper around the z-axis is calculated using real-world coordinates. Rotations around the x- and y-axes are zero.

3.7. Grasp-RCNN Overview

A global overview of the grasp prediction sequence is visualized in fig. 3.2. On the left, we got an input RGB image that is fed to the network. The Mask-RCNN gives as output two object masks (in yellow) and the Faster-RCNN predicts around twenty output grasps (in green). The post-processing step computes for each grasp the grasp score, sorts the grasps based on the object height and grasp score, and checks if the grasp is in a collision. In this scenario, the object on the right is the highest, so the best grasp around this object is selected (in blue). The pixel coordinates are transformed into real-world coordinates and the object is grasped.



(a) Input RGB data (b) Output from the network (c) Grasp in pointcloud

Figure 3.2. The RGB image (a) is fed into the network, the network predicts (b) masks for the objects in yellow, grasp rectangles (represented as green lines), and their respective bounding boxes (orange). The post-processing creates a score for each grasp and prioritizes them on (1) the highest objects and (2) the highest grasp score. (top 1 grasp presented as a blue line). In (c) the grasp is seen in a 3D visualization.

In the next chapter, the experimental results are discussed.

This page is intentionally left blank.

4

Experiments and Results

The grasp performance of Grasp-RCNN is evaluated by robot experiments. During the experiment, 35 different objects of various sizes, shapes, materials and colors were picked. A total of around 1900 grasp attempts were made both in the single object and in cluttered scenes. A total of three different experiments are conducted.

4.1. Setup

The robot platform setup is shown in fig. 4.1 and consists of a UR5 robot manipulator. As an end-effector, a Robotiq parallel plate gripper is used. The vision hardware consists of an RGB sensor from IDs and a depth camera by Ensenso. For the experiments, a set of 20 known objects and 15 unknown objects is used with a variety of shapes, colors, materials, and sizes. Objects are chosen such that at least one of the dimensions does not exceed the maximum width of the gripper. Complex shapes such as a nerf gun or a bath duck were used to test how the network deals with complex objects. The objects are shown in figs. B.1 and B.2. During the different experiments, objects are randomly placed in the scene.

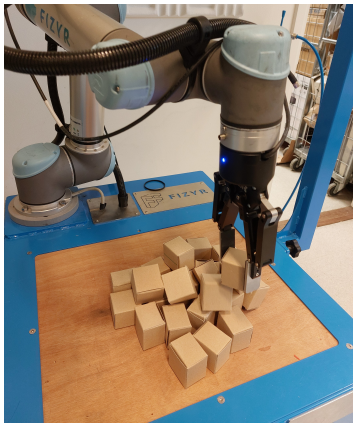


Figure 4.1. The UR5 is performing a pick with the Robotiq parallel plate gripper. The RGB and depth cameras are located above the pick area.

4.2. Metrics

To analyze the performance two metrics are used: success-rate and completion-rate. The success-rate is the number of successful grasps as a percentage of the total amount of attempted grasps. A grasp is considered successful when the robot lifts the object and moves it away from the gripping area to a predefined location without collision. An attempt is started at the moment the network takes a new photo of the scene. A scene where no grasps can be found by the network is seen as a failed attempt. The second metric is the completion-rate: this is the percentage of scenes in which a table is fully cleared by the robot. In single-object experiments, the success-rate and completion-rate are the same.

During the experiments, it is observed that in some cases the gripper collides with the object that it is going to grasp, but still succeeds in doing so. For single object scenes (Experiment 1), these grasps-with-collision attempts are discarded as a failure. For multiple object scenes (Experiments 2 and 3), these types of grasp-with-collision are discarded, and therefore seen as failures in the success-rate. However, the gripper is able to grasp and remove the object from the scene. Therefore a scene where one or multiple grasp-with-collision events occur can still be completed, and thus do count for the completion-rate.
This same strategy is used for grasps that pick multiple objects. In some cases, the gripper picked up two objects, instead of one. This could occur in cases where the robot could not make a distinction

between the objects, or when the robot picked up the lowest of the stacked objects. In these cases, the grasp is discarded as a failure for the success-rate, but the grasp does count for the completion-rate.

4.3. Experiment Methodology

To test Grasp-RCNN a total of three experiments are conducted. The first experiment is to test the network on single objects in the scene. Each object is grasped in 20 different scenes. In these scenes, the objects are manually placed in random locations and with random orientations to ensure that all the orientations of the object are present.
The second experiment is to test the performance of the method in a cluttered scene. The experiment procedure is as follows: **(1)** Choose 8 of the 35 objects at random and put them in the box and turn the box around on the table to create a random cluttered scene; **(2)** The robot attempts multiple grasps until all objects are grasped or three consecutive grasp attempts failed. The objects are chosen at random but it is checked that all the objects are equally represented in the cluttered scenes.
In the third experiment, a simulation is made of a bin-picking scenario where a bin must be emptied. We simulate this kind of environment by putting 40 similar boxes on the table. The goal of this experiment is to pick as many objects as possible. The robot attempts multiple grasps until all objects are grasped, or three consecutive grasp attempts fail.

4.4. Experiment 1 - Single objects

During the test of the network on single objects, the orientation of the object was varied such that it was present in different orientations. For objects such as the superglue, attempts are made both in the standing and lying positions. During the experiment, 35 objects were grasped between 18 and 22 times with a total of 698 grasps attempts. In table 4.1 the number of grasp attempts and the success-rate of the single object experiments are shown. The algorithm can achieve an overall success-rate of 89.68% for grasping single objects in a scene, with a total average processing time of 616ms. Split into known and unknown objects a success-rate of 87.73% and 90.94% respectively is achieved. An overall success-rate of 89.7% is achieved.

The success-rate for unknown objects is higher than that of known objects, while a neural network normally achieves better on known objects. This can partly be explained by the number of unknown objects that can be described as box-type objects, which can be grasped more easily. Another explanation is some bad-performing known objects. In the known objects group there are four bad performers; the spatula, the white piggy-bank, the nerf-gun and the wooden dear, while in unknown object group has only one bad performing object, the orange building block. From three of these objects, it is observed that their bad performance is likely to be dependent on their shape. In section 4.7, these objects and their typical failures are further discussed.

Table 4.1. Number of attempts and the Success-Rate (S-R) of objects in Experiment 1. The objects are categorised in three categories: box-shaped, round or cylindrical shaped, and complex shaped. The object known from the training are shown in the top-part of the table, the bottom part contain the unknown objects. The average success-rate under each column is a weighted average.

Known objects

Box-shaped			Round / Cylindrical shapes			Complex Shapes		
Objects	Attempts	S-R	Objects	Attempts	S-R	Objects	Attempts	S-R
Grey box	21	90.5	Glue Pen	19	94.7	Wooden Dear	20	75
White medicine box	20	95	Nail polish	20	95	Bath Duck	21	85.7
Container (green)	20	85	Cup	20	95	Piggy bank color	20	95
Stamper	20	100	Spatula	20	60	Piggy bank white	20	65
Perforator (purple)	20	100	Paint Roller	20	90	Styrofoam cat	20	95
			Totem Pole	20	85	Wooden tree	19	94.7
			Super Glue	20	90	Nerf-gun	20	70
						Tennis balls	20	95
Average		94.1	Average		87.0	Average		84.4
Average Known		87.73						

Unknown objects

Box-shaped			Round / Cylindrical shapes			Complex Shapes		
Objects	Attempts	S-R	Objects	Attempts	S-R	Objects	Attempts	S-R
Brown Camera Box	20	95	Tomato	20	95	Banana	19	100
Orange Building Block	20	65	Pipe	20	90	X-Box controller	18	94.4
Red house	20	95	Triangular Box	20	95	Nut cracker	20	85
Glasses Case (wood)	21	90,5	Stuffed Wood Pecker	20	100	Peeler	21	81
Stapler (green)	20	100				Headphones	20	95
Measuring Tape	19	84,2						
Average		88.63	Average		95.0	Average		90.8
Average unknown		90,94						
Average Total		**89.68**						

4.5. Experiment 2 - Cluttered Scenes

To test the ability of the algorithm to handle cluttered scenes, a total of 81 cluttered scenes were created with a total of 694 grasp attempts. Each start scene consists of eight objects (fig. 4.2) chosen at random from the objects used in Experiment 1. The algorithm achieves an 81.0% success-rate in cluttered scenes and an 84.6% completion rate with an average process time of 739 ms per scene.

From these results, it can be seen that the success-rate of Experiment 1 drops from 89.7% to 81.0%. This is caused to the increase in the number of collisions and rejected grasps. A grasp is rejected if the algorithm does not find a feasible grasp during the prediction or post-processing stage. When more objects are present in the scene, the probability that objects are close to each other gets higher. A more detailed overview of the different types of failures is given in section 4.7.

Figure 4.2. For the cluttered scenes experiment, 8 out of the 35 objects are put into a box, where after the box is turned on the table, creating cluttered scenes.

A second observation from the results is that the completion-rate is higher than the success-rate. In section 4.3 it is explained that grasps-with-collision are not incorporated for the success-rate but are incorporated for the completion-rate. During eight grasp attempts the gripper picked up multiple objects, and with 14 attempts a grasp-with-collisions occurred. This is shown in table 4.3.

4.6. Experiment 3 - Bin picking boxes

In the multiple-box experiment, a total of 17 start scenes are generated with a total of 502 grasps attempts. A total of 439 collision-free grasps were conducted resulting in a success-rate of 87.5% with an average process time of 1235 ms per scene. In 8 out of 17 scenes, the gripper was able to empty the table. In the 9 other cases, the network failed to find feasible grasps on three consecutive tries. This resulted in a completion-rate of 47.1%. The big difference between the success-rate and completion-rate can be explained by the grasp streak during the experiment. During this experiment, the system could grasp on average 8.7 boxes before a failure. The longest consecutive grasp streak without failure was 51 grasps. During the experiments, it was observed that the network could predict and commence grasps not only at the border of the clutter but also when holes were present where the gripper would fit in.

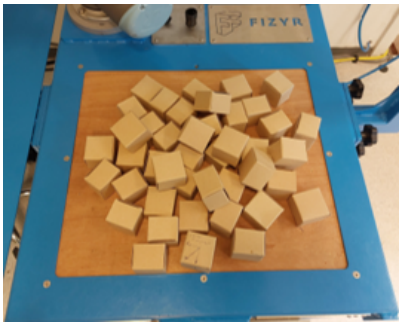


Figure 4.3. A start scene for the bin-picking experiment with similar boxes. A total of 17 similar scenes are created.

Three of the nine bins, the system was not able to clear, are shown in fig. 4.4. The objects are densely packed and are stacked in two layers. Looking from a 2D perspective, it is hard to find a collision-free antipodal grasp location. However, if viewed from a 3D perspective, some objects are possible to grasp. These possible grasps were not seen by the network due to design choices made in the grasp generation tool. Stacked objects were only grasped when both ends of the grasp rectangle were above the table, instead of on top of other objects.

Table 4.2. The success-rate and completion-rate for the different experiments.

	Single objects	Multiple objects	bin-picking
Success-rate	89.7	81.0	87.5
Completion-rate	-	84.6	47.1

Figure 4.4. In experiment 3, the algorithm was not able to generate feasible grasps for the nine scenarios, three of them are shown.

4.7. Grasp failures

During the three grasp experiments, different kinds of grasp failures were observed. The gripper grasped at the wrong height, collided with (other) objects, or the network was not able to predict a grasp at all. This section elaborates on different failure types and the per-object failures.

4.7.1. Failure types

To get an understanding of the strong and weak points of the model, the failed grasp attempts are summarized and shown in table 4.3. From the failure types in the table it can be concluded that with a cluttered scene, e.g. more objects in a scene, the amount failures due to collisions and the number of not-feasible-grasps increased.
No-feasible-grasps occur when Grasp-RCNN is unable to predict a grasp in a scene or when the post-processing rejects all grasps due to collisions or lack of overlap.
Failures based on wrong finger distance are errors in the predictions made by the Grasp-RCNN itself. A wrong finger distance could occur in two different ways. The first option is that both the grasp and the object are predicted smaller than they are. The smaller grasp is sent to the post-processing algorithm which should find that there is not enough overlap. In case of an error in the height data at both fingers, the grasp could still be conducted. The second option and more likely explanation is that a correct grasp is sent to the post-processing, but that due to camera distortion, the coordinates are made smaller than they should be.
The errors due to NaN values occur when there are not enough pointcloud points at the location of the fingers or at the gripper center. This is caused by occlusions due to higher objects, which are more likely to occur in cluttered scenes.

Failures based on heights, hollow objects or slips are discussed in the next section.

Table 4.3. An overview of the different reasons of grasp failure of single object experiments (Exp. 1), multiple objects experiments (Exp. 2) and bin-picking experiments (Exp.3) For some objects, failures are very specific. These objects are noted with the number of described failures during experiment 1 and 2. For collision a distinguish is made between collision with and without a grasp.

Reason of failure	Exp. 1	Exp. 2	Exp. 3	Specific objects
Total	72	125	63	
Collision (w. / w.o grasp)	10 / 1	14 / 9	15 / 7	-
Wrong height	6	10	0	White Piggy-bank [3/3]
Wrong finger distance	5	9	0	-
Hollow objects	12	10	0	Orange Block [4/1], Peeler [4/9]
Slip of object	10	9	0	Spatula [6/7], Nerfgun [3/1]
No feasible grasp predicted	9	17	29	-
NaN values at grasp	5	12	0	-
Not specified failures	14	26	7	-
Grasp of multiple objects	-	8	5	-

4.7.2. Per-object failures

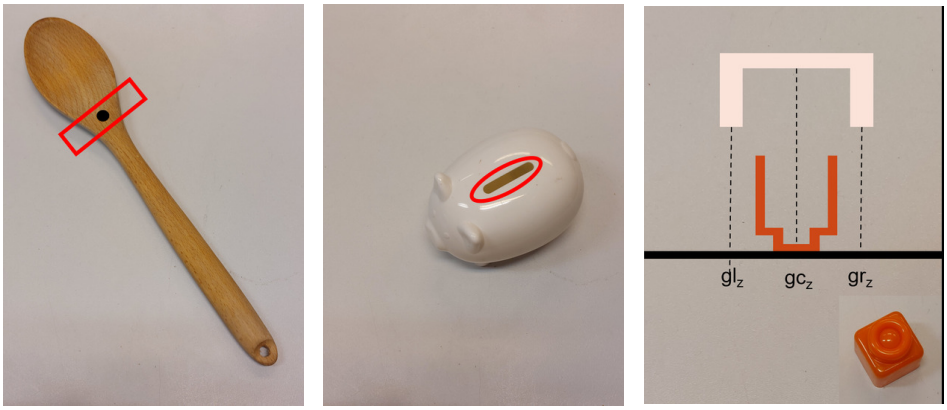
From table 4.1 it can be observed that the success-rate of some objects is under-performing. For some of these objects, the failed attempts are very specific and caused by specific reasons. Looking at the performance on a per-object basis can reveal what types of failures occur in the algorithm. The most difficult items to pick, are items with the lowest success-rate.

Of the known objects, the spatula, white piggy bank, wooden dear, and nerf-gun are achieving a success-rate below 75%. For the unknown objects, only the orange building block has a low success-rate of 65%. In fig. 4.5, objects are shown with a low performance together with indications that cause their failures.

The spatula's (fig. 4.5a) CoM is located just on the edge of the haft. A predicted grasp on top of the CoM, as indicated with the black dot and red gripper representations, is not antipodal. The edges are inclined and the object will slip through the fingers of the gripper.

For the piggy bank, (fig. 4.5b), failures occur when the center of the gripper is predicted on top of the coin slot. The height of the object at the gripper's center could therefore not be estimated by the post-processing, causing the gripper to approach the object to high.

The low performance of the orange block is caused by the scenes in which the block is upside down (fig. 4.5c). The algorithm detects the object and predicts the grasp, but the difference in the overlap between the center and the left and right gripper is not sufficient enough to go for a grasp. In the dataset, two more objects have a hollow structure: the cup and the green container. However, when the cavity of these objects is facing up, the algorithm finds a feasible grasp at the ear of the cup, or the sides of the green container, see fig. 4.6



(a) (b) (c)

Figure 4.5. These objects have a low success-rate during the single object experiments. Grasp fail due to slipping on a grasp around the CoM (4.5a), a grasp attempt with wrong height estimation due to the coin-slot (4.5b) and no feasible grasp due to a lack of overlap when objects are hollow (4.5c).

(a) (b)

Figure 4.6. Grasp-RCNN is able to find grasps for hollow objects. A successful grasp of (a) a container at the side and (b) a coffee cup at the ear.

4.8. Prediction and post-processing time

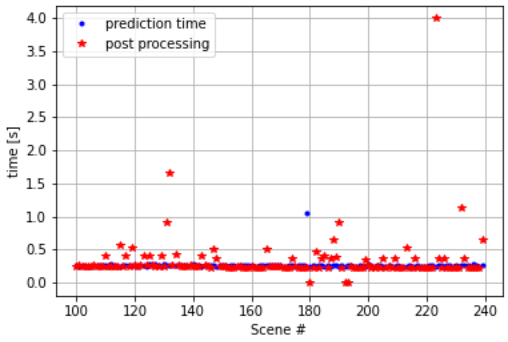
In automation processes, the total prediction time is an important factor. If the prediction time exceeds the movement time of the robot, the robot has to wait before it can pick a new object. In the experimental phase, the prediction and movements are in sequence. When the current method will be used in production, new images are made when the robot is moved outside the grasp area.
In table 4.4, the average prediction time, post-processing time, and total process time are given. It is seen that the average time for both prediction and post-processing goes up in between experiments. This suggested that the computation time is dependent on the number of objects, observed by the network, in the scene.
It is worth mentioning that for the multiple object experiment the average times also include single object scenarios. If the number of objects would be kept on a higher constant value, the average would increase.

Table 4.4. The average prediction- and post-processing time of the different methods. The prediction and post-processing are done in sequence, so the total process time is the total waiting time of the robot.

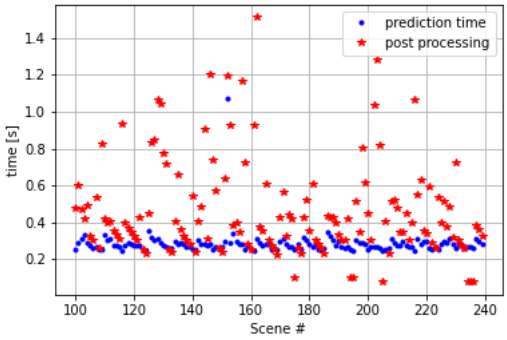
	Single object [ms]	Multiple objects [ms]	bin-picking [ms]
Average prediction time	278.3	287.9	392.7
Average post-processing time	337.6	450.9	838.9
Total process time	615.9	738.8	1235.2

In fig. 4.7 the post-processing and prediction time are shown for 140 consecutive scenes (scene number 100 to number 240) in all three experiments. For the single object experiments (fig. 4.7a), a clear constant prediction- and post-processing time is seen. However, looking at figs. 4.7b and 4.7c, a tooth profile is visible caused by a changing number of objects in the scene. With more objects in the scene, more object segmentation and grasps are predicted, which takes time. The tooth profile is more visible in the post-processing time with a duration between around 250 up to 1000+ ms than with the prediction-time which lies between 250 and 380 ms.

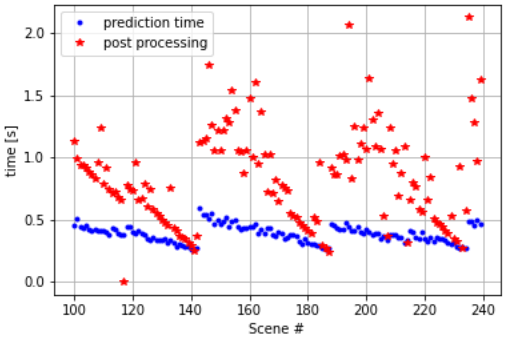
To understand the dependence of the time on the number of objects, the prediction, post-process, and total process time are shown against the number of objects in the scene. This is done separately for experiments 2 and 3. It can be seen that an upward trend for both the prediction and post-processing time *vs.* the number of objects exist.



(a) Experiment 1: Single Objects scenarios

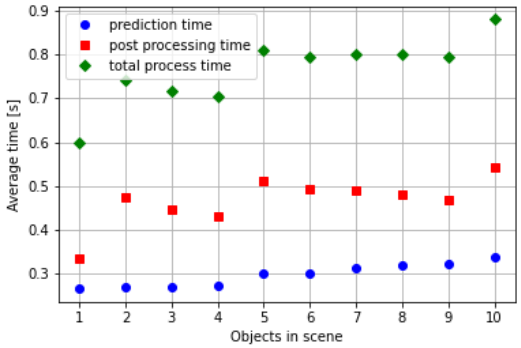


(b) Experiment 2: Multiple objects scenarios

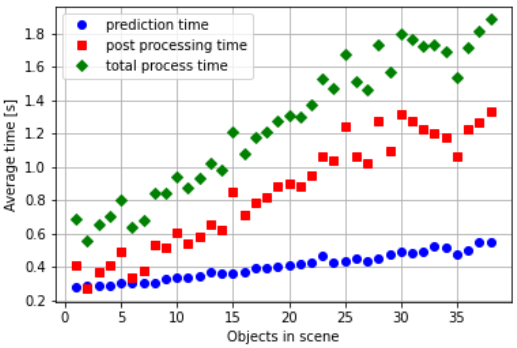


(c) Experiment 3: Multiple Boxes scenarios

Figure 4.7. The post-processing and prediction throughout the experiment. For visualization reasons, only scene 100 to 240 are shown. It can be seen clearly that in the single object scenes (fig. 4.7a) the time stays constant. In the multiple-object (fig. 4.7b) and bin-picking scenarios (fig. 4.7c) decay in the post-process time and prediction time is seen. This is caused by a decreasing amount of objects, thus fewer calculations.



(a) Duration of the prediction and post-processing for Experiment 2.



(b) Duration of prediction and post-processing for Experiment 3.

Figure 4.8. The average prediction, post-processing and total-process time of Experiments 2 and 3. As the number of objects in the scene increase, the total-process time also increases.

5 Discussion

To obtain the results of this research some assumptions and simplifications were made which can affect the performance of Grasp-RCNN. The assumptions and simplifications made in each part of the pipeline are discussed in their respective section.

5.1. Data Generation

The grasp generation is the first part of the grasp pipeline. Assumptions, simplifications and design choices in the grasp generation affect the way the network is trained and what output the network will give. In this section, the grasp annotation tool is discussed.

5.1.1. Used Dataset

In this thesis, one existing dataset from Fizyr is used for the grasp generation consisting of 765 images with 66 individual objects [32]. A total of 3883 objects were captured and 102275 grasps were generated. In terms of machine learning networks, this is a quite small dataset and increasing the dataset size could positively influence the prediction of Grasp-RCNN. As discussed in the introduction, some grasp databases exist that could extend this dataset, but it is chosen not to use them during this thesis for several reasons. [14, 18] created both big datasets with 800k and 6.7M grasps images respectively. However, these datasets only contain one grasp per object and were therefore not used. This could lead to issues when attempting to grasp objects in cluttered scenes.
Two grasp datasets are present that contain multiple grasps per object: the Cornell Grasp Dataset [9] and the Jacquard Database [5]. The Cornell Dataset consists of 885 images and pointcloud with a total of 240 objects and 8019 hand-labeled grasps, which is still a small dataset in terms of machine learning data. Therefore [5] created the Jacquard Database that consists of 54k images and 1.1M grasps based on an automatizing algorithm. Due to the large variety of objects in the Jacquard dataset, this should generalize to a variety of objects. For this thesis, this could have helped to give the network a ”warm-start” for predicting grasps, before training the network on cluttered scenes. However, this database is only available for academic purposes and due to the collaboration with Fizyr, could not be used.

5.1.2. Grasp collisions

The generated grasp needs to pick up objects without colliding with objects or the environment. In the collision detection phase (section 2.5.2) grasps were labeled ”in collision” and discarded when one of the ends of the grasp rectangle had overlap with the grasped object or with other object masks. Therefore only grasp rectangles where the fingers were above the table were seen as correct. As consequence, for a high object that is surrounded by lower objects, no grasps are generated. Also when objects are stacked and the bottom objects are still seen by the network, the annotation tool does not generate feasible grasps. This caused the network not to learn these kinds of grasps. In Experiments 2 and 3 this resulted in higher ”no feasible grasp” failures, while grasps were feasible, which let to a lower success- and completion-rate.

5.2. Grasp-RCNN

Grasp-RCNN consists of two independent networks: one to predict grasps (Faster-RCNN) and one to predict objects (Mask-RCNN). The networks share the same architecture but predict different types of data, grasps and objects. Since multiple grasps are predicted per object, the Region Proposal Network cannot be shared. However, they could have shared the first set of CNN layers. This could decrease the computation time during both the training and prediction phases. If this measure also increases the accuracy of the network needs to be investigated. It is possible that the shared features maps make both parts of the network more aware of good grasps. However, as the features learned for the object detection task could be fundamentally different than the features learned for the grasp estimations option, this implementation should be tested first.

5.3. Post-Processing

The grasps that are predicted by the network are passed through to the post-processing part. This section describes the assumptions made in the post-processing that could impact the performance.

5.3.1. Pixel *vs.* real-life location

The Grasp-RCNN predicts grasps on a 2D image in pixel coordinates. In the post-process, the grasp representation is transformed from pixel coordinates to real-world coordinates. In post-processing, the camera view is assumed to be parallel, instead of having a cone shape. Therefore the distortion of the camera is not taken into account, which can lead to incorrect real-world finger locations. This effect occurs mainly at the side of the image. For multiple-object scenes, where objects are close together and near the border of the image, this could lead to collisions.

5.3.2. Speed

The current network predicts a grasp on average of 616 ms in single object environments and 739 ms in multi-object environments. Of the total process-time, the post-processing grows fast when the number of objects increases. When the robot speed goes up, and the prediction and grasping are done in parallel, with the current process-time it could happen that the robot has to wait for the prediction. This can be very costly and should be avoided.

5.4. Recommendations and Future Research

This section contains recommendations that should result in an improvement of the network.

5.4.1. Larger Dataset

Improve the current dataset by adding a wide variety of objects in single and multiple object scenarios. With a bigger dataset, the network can generalize better on new images and objects. The current pipeline needs (hand) annotated object masks, which can be time-consuming. Another possibility is to create a synthetic database such as done by [5] with the Jacquard Dataset. This database should be composed of a simulation environment that automatically generates grasps. With this step, one should be aware of the simulation/real-life learning gap. In literature solutions to bridge this synthetic to real-life data gap are proposed [1, 29]. A possible hybrid solution between real-life images and synthetic images could also work.

5.4.2. Better collision detection

In the current method possible feasible grasps are removed during the collision detection of the grasp generation. It is advised to change the collision detection in such a way that grasps are not directly removed when the fingers overlap with other object masks, but use depth information to check if enough overlap exists to grasp the object without colliding with its surrounding. This should result in better grasp generations which lead to better grasp predictions in densely cluttered scenes, or with stacked objects.

5.4.3. Add depth information in the network

In line with the previous recommendation, It could be beneficial to add depth information to the neural network. With the current setup, this could be done by removing the blue channel and replacing it with

the depth information from the pointcloud data. Together with improved collision detection, this should result in more feasible grasps because the network is more aware of its environment

5.4.4. Increase speed

If the speed of prediction is to slow for a production pipeline, the following improvements can be made. First, the CNN of the Mask-RCNN and Faster-RCNN should be combined, such that the object and grasp network share the same basis. In the current method, the networks predict in series, while they could predict in parallel. Another option would be to incorporate the grasp-scores already in the object-annotation and feed this to the Grasp-RCNN network. The network is then trained on preferred grasps, and the whole object and grasp matching could be removed from the post-processing part.

5.4.5. Shaker Plate

Regardless if changes are made according to the recommendations in the software, it could positively influence the performance in cluttered scenes if a shaker plate is attached to the table or bin. In this way, when the network does not find a feasible grasp, the scene could be shaken up, changing the locations of objects and thus creating a different scene.

This page is intentionally left blank.

6

Conclusion

This thesis worked towards the objective to create a bin-picking pipeline that uses deep learning to generate a complete solution for bin-picking with a two-fingered gripper. The grasp pipeline consists of a grasp-generation method, a neural-network architecture, Grasp-RCNN, and a decision algorithm. The grasp-annotation tool generates a wide variety of grasps that are antipodal and collision-free based on an RGB image and a pointcloud. With these generated grasps Grasp-RCNN is trained. Grasp-RCNN consists of two independent networks for object and grasp detection followed by a decision algorithm. The network can predict feasible grasps that are processed by the decision algorithm to select the best grasp. Physical experiments are performed and show that Grasp-RCNN can pick both known as unknown objects. The proposed algorithm achieves an 89.68% and 81.0% success-rate with an average total-process time of 616 ms and 739 ms in single and multiple object scenarios, respectively. The dataset contains both known and unknown objects. In bin-picking experiments, a success-rate of 87.5% with a total process time of 1235 ms is achieved. These results indicate that the accuracy of the proposed grasp-network is comparable to the state-of-the-art, but that the speed of the network still can be improved.

This page is intentionally left blank.

Bibliography

[1] Tariq Alkhalifah, Hanchen Wang, and Oleg Ovcharenko. *MLReal: Bridging the gap between training on synthetic data and real data applications in machine learning*. 2021. arXiv: 2109.05294 [physics.geo-ph].

[2] Sai Balaji. *Binary Image classifier CNN using TensorFlow*. 2020. URL: https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697.

[3] I-Ming Chen and Joel W. Burdick. "Finding Antipodal Points Grasps on Irregularly Shaped Objects". In: *IEEE Transactions on Robotics and Automation* 09.4 (1993), pp. 507–512.

[4] Amaury Depierre, Emmanuel Dellandr'ea, and Liming Chen. "Scoring Graspability based on Grasp Regression for Better Grasp Prediction". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), pp. 4370–4376.

[5] Amaury Depierre, Emmanuel Dellandréa, and Liming Chen. "Jacquard: A Large Scale Dataset for Robotic Grasp Detection". In: *CoRR* abs/1803.11469 (2018). arXiv: 1803.11469. URL: http://arxiv.org/abs/1803.11469.

[6] Di Guo et al. "A hybrid deep architecture for robotic grasp detection". In: (2017), pp. 1609–1614. DOI: 10.1109/ICRA.2017.7989191.

[7] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: (2015). DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

[8] Kaiming He et al. "Mask R-CNN". In: *CoRR* abs/1703.06870 (2017). arXiv: 1703.06870. URL: http://arxiv.org/abs/1703.06870.

[9] Yun Jiang, Stephen Moseson, and Ashutosh Saxena. "Efficient grasping from RGBD images: Learning using a new rectangle representation". In: *2011 IEEE International Conference on Robotics and Automation* (2011), pp. 3304–3311.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[11] S. Kumra and Christopher Kanan. "Robotic grasp detection using deep convolutional neural networks". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017), pp. 769–776.

[12] Dive Into Deep Learning. "Convolutional Neural Networks - Convolution For images". In: (). [Online; accessed 02-08-2022]. URL: https://d2l.ai/chapter_convolutional-neural-networks/conv-layer.html%5C#convolutional-layers.

[13] I. Lenz, Honglak Lee, and Ashutosh Saxena. "Deep learning for detecting robotic grasps". In: *The International Journal of Robotics Research* 34 (2015), pp. 705–724.

[14] Sergey Levine et al. "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection". In: *CoRR* abs/1603.02199 (2016). arXiv: 1603.02199. URL: http://arxiv.org/abs/1603.02199.

[15] Tong Li et al. "Keypoint-Based Robotic Grasp Detection Scheme in Multi-Object Scenes". In: *Sensors* 21.6 (2021). ISSN: 1424-8220. URL: https://www.mdpi.com/1424-8220/21/6/2132.

[16] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2014. DOI: 10.48550/ARXIV.1405.0312. URL: https://arxiv.org/abs/1405.0312.

[17] Jens Lundell, Francesco Verdoja, and Ville Kyrki. "Robust Grasp Planning Over Uncertain Shape Completions". In: *CoRR* abs/1903.00645 (2019). arXiv: 1903.00645. URL: http://arxiv.org/abs/1903.00645.

[18] Jeffrey Mahler et al. *Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics*. 2017. DOI: 10.48550/ARXIV.1703.09312. URL: https://arxiv.org/abs/1703.09312.

[19] Jeffrey Mahler et al. *Dex-Net 3.0: Computing Robust Robot Vacuum Suction Grasp Targets in Point Clouds using a New Analytic Model and Deep Learning*. 2017. DOI: 10.48550/ARXIV.1709.06670. URL: https://arxiv.org/abs/1709.06670.

[20] Timothy Patten, Kiru Park, and Markus Vincze. "DGCM-Net: Dense Geometrical Correspondence Matching Network for Incremental Experience-Based Robotic Grasping". In: *Frontiers in Robotics and AI* 7 (2020). DOI: 10.3389/frobt.2020.00120. URL: https://doi.org/10.3389%2Ffrobt.2020.00120.

[21] Lerrel Pinto and Abhinav Gupta. "Supersizing Self-Supervision: Learning to Grasp from 50K Tries and 700 Robot Hours". In: (2016), pp. 3406–3413. DOI: 10.1109/ICRA.2016.7487517. URL: https://doi.org/10.1109/ICRA.2016.7487517.

[22] Joseph Redmon and Anelia Angelova. "Real-Time Grasp Detection Using Convolutional Neural Networks". In: (2015).

[23] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

[24] Robotiq. *Instructions Manual — Specification*. [Online; accessed 18-July-2022]. URL: https://assets.robotiq.com/website-assets/support_documents/document/online/2F-85_2F-140_TM_InstructionManual_HTML5_20190503.zip/2F-85_2F-140_TM_InstructionManual_HTML5/Content/6.%5C%20Specifications.htm.

[25] Ashutosh Saxena et al. "Robotic Grasping of Novel Objects". In: *NIPS*. 2006.

[26] scikit developers. *Skeletonize*. [Online; accessed 03-July-2022]. 2022. URL: https://scikit-image.org/docs/stable/auto_examples/edges/plot_skeleton.html#zha84.

[27] Hao Tian et al. "Transferring Grasp Configurations using Active Learning and Local Replanning". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 1622–1628. DOI: 10.1109/ICRA.2019.8793796.

[28] Vinicio Tincani et al. "Velvet fingers: A dexterous gripper with active surfaces". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 1257–1263. DOI: 10.1109/IROS.2012.6385939.

[29] Jonathan Tremblay et al. *Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization*. 2018. arXiv: 1804.06516 [cs.CV].

[30] Jacob Varley et al. "Shape Completion Enabled Robotic Grasping". In: *CoRR* abs/1609.08546 (2016). arXiv: 1609.08546. URL: http://arxiv.org/abs/1609.08546.

[31] Mohit Vohra, Ravi Prakash, and Laxmidhar Behera. "Real-time Grasp Pose Estimation for Novel Objects in Densely Cluttered Environment". In: *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)* (2019), pp. 1–6.

[32] Mark Wiersma. *Map-Grasp; A Deep-Learning approach to bin-picking*. 2021. URL: http://resolver.tudelft.nl/uuid:5f40a888-ab4b-4de1-b1ce-2d9f8413d003.

[33] Wikipedia contributors. *Principal component analysis — Wikipedia, The Free Encyclopedia*. [Online; accessed 22-June-2022]. 2022. URL: https://en.wikipedia.org/wiki/Principal_component_analysis.

[34] Muhamad Yani, S Irawan, and Casi Setianingsih. "Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail". In: *Journal of Physics: Conference Series* 1201 (May 2019), p. 012052. DOI: 10.1088/1742-6596/1201/1/012052.

[35] Andy Zeng et al. *Multi-view Self-supervised Deep Learning for 6D Pose Estimation in the Amazon Picking Challenge*. 2016. DOI: 10.48550/ARXIV.1609.09475. URL: https://arxiv.org/abs/1609.09475.

[36] Andy Zeng et al. “Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching”. In: *CoRR* abs/1710.01330 (2017). arXiv: 1710.01330. URL: http://arxiv.org/abs/1710.01330.

[37] Hanbo Zhang et al. “ROI-based Robotic Grasp Detection for Object Overlapping Scenes”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 4768–4775. DOI: 10.1109/IROS40897.2019.8967869.

[38] T. Y. Zhang and C. Y. Suen. “A Fast Parallel Algorithm for Thinning Digital Patterns”. In: *Commun. ACM* 27.3 (Mar. 1984), pp. 236–239. ISSN: 0001-0782. DOI: 10.1145/357994.358023. URL: https://doi.org/10.1145/357994.358023.

[39] Guoliang Zhong, Yangdong Hou, and Weiqiang Dou. “A soft pneumatic dexterous gripper with convertible grasping modes”. In: *International Journal of Mechanical Sciences* 153-154 (2019), pp. 445–456. ISSN: 0020-7403. DOI: https://doi.org/10.1016/j.ijmecsci.2019.02.028. URL: https://www.sciencedirect.com/science/article/pii/S0020740318323324.

[40] Xinwen Zhou et al. “Fully Convolutional Grasp Detection Network with Oriented Anchor Box”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), pp. 7223–7230.

This page is intentionally left blank.

A

CNN and RPN

In chapter 3 the Grasp-RCNN architecture is explained. Grasp-RCNN model consists of two separate networks, one for detecting grasps (Faster-RCNN) and one for detecting objects (Mask-RCNN). Faster-RCNN and Mask-RCNN networks are in the base the same. Within Mask-RCNN, an extra branch is added that predicts the object segmentation. For computer vision, the RCNN networks are widely used for object detection and segmentation. RCNN networks are CNN networks that look at particular regions, instead of the whole data.

A.1. Convolutional Neural Network - CNN

A Convolutional Neural Network (CNN) is a neural network that takes an image as input and can differentiate between objects in the image. Therefore they are widely used and very effective for recognition and classification tasks in images. A CNN consist of two main parts; A feature extraction part and a classification part. These parts are built up from convolutional layers, pooling layers and fully connected layers. An overview is shown in fig. A.1

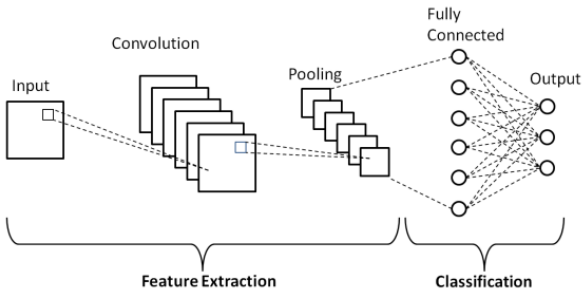


Figure A.1. An overview of CNN Network. A CNN consists of multiple stacked feature extractions followed by one classification layer. (image from: [2])

A.1.1. Convolutional Layer

A convolutional layer is used to extract the various features from the input images based on a method called convolution. With a convolution spatial information of the image is extracted by applying kernel filters to the input image. During the forward pass of the training, the kernel slides across the image creating a representation of the image. The kernel is a matrix of trained parameters and shares the local properties spatially.

A.1.2. Pooling layer

The pooling layer is responsible for reducing the spatial size of the feature maps generated by the convolution layer. This decrease the amount of data, and therefore the computational costs that are required. However, within the pooling layer, dominant features that are rotation or positional invariant are still extracted. This makes the network not biased toward small changes within the image.

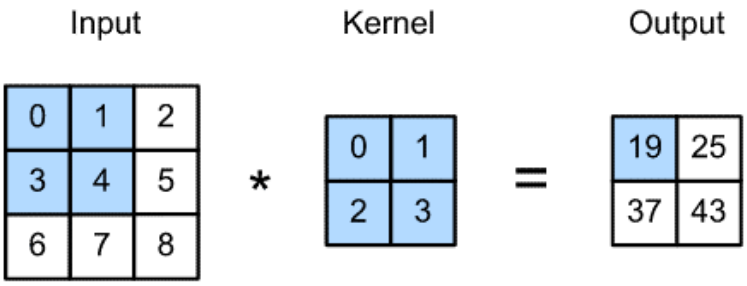


Figure A.2. A convolutional of an image with a kernel. The kernel slides across the image creating a new representation of the image. (image from: [12])

Two types of pooling are commonly used: Averaged pooling and max pooling. Average pooling returns the average value from the patch covered by the kernel, and max pooling is the maximum value. According to literature, max pooling performs a lot better than average pooling because it removes the noise from the convolutional activation functions. Figure A.3 visualizes the pooling process.

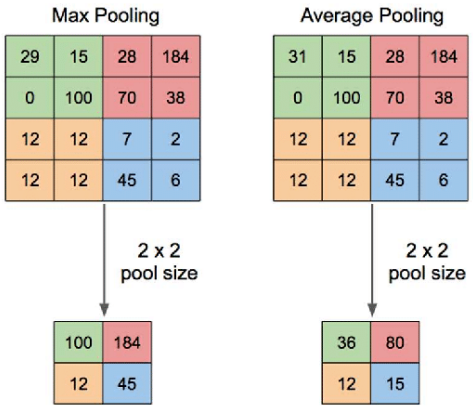


Figure A.3. A max pooling and average pooling operation with a 2x2 pool. At max pooling, the maximum value of each pool is taken, while with average pooling the average is taken. (image from: [34])

A.1.3. Fully Connected Layer

After several consecutive layers of the convolutional and pooling layers, a classification of the image is needed. For this process, the output of the last pooling layer is flattened and used as input for the Fully Connected Layer (FC-layer). In several consecutive FC-layers, all the neurons are connected to the previous and next layers of the network. The last layer of this FC-layer stack is the output layer, which gives the result of the classification.

A.2. Region proposal network

The RCNN of Faster-RCNN and Mask-RCNN stands for Region-based Convolution Neural Network (R-CNN) and is an architecture for object detection. A R-CNN uses the in appendix A.1 described CNN, without the fully connective layers, and a region proposal algorithm that generate bounding boxes or location proposals of possible objects in the image. The RPN, consists of a classification layer and an added regression branch to obtain better values for the bounding boxes and for the anchors.

A RPN takes anchor as input and has as output a set of rectangular region proposals together with an objectness score. The region proposals are generated by sliding a small network over the feature map of the last convolutional layers. The input of this network is a $n \times n$ spatial window of the convolutional feature map, and is mapped to a lower dimensional feature map. This lower dimensional feature is fed into two FC-layers, one for classification and one for regression. Because of the sliding window

approach, the FC-layers are shared across all the spatial locations. At each location of the sliding window, multiple region proposals are predicted called anchors. For each anchor, four regressions outputs are given: the coordinates of the anchor, and classification is given: background or no background. The anchors are centred at the sliding window and are proposed with multiple scales and aspect ratios. In this way, objects of different sizes and shapes can be detected. The result of this approach is that the network is invariant to translation. This means that the same object should be predicted across the image.
The predicted bounding boxes with a foreground classification are used as input for the Region of Interest (RoI) pooling layer. This layer takes two inputs: A fixed-size feature map from the CNN with multiple convolutions and max pooling layers, and a list of $4k$ with the coordinates of the RoI bounding boxes with k as the number of RoI's. [23]
A typical CNN is only able to classify an object and regress its bounding box for one object at a time. When multiple objects are present, the bounding box regression will not work. Therefore a Region Proposal Network is used to decide which smaller regions of the image are "worth" looking into, to reduce the computational requirements of the overall inference process. This way, the RCNN forces the CNN to focus on only these small regions. The output of a RPN are bounding boxes, that will be passed to a classifier and regressor to check the occurrence of objects. The RPN predicts thus the possibility of an anchor being a background or a foreground.

This page is intentionally left blank.

B

Grasp objects

Known objects

Figure B.1. Known objects during the experiments.

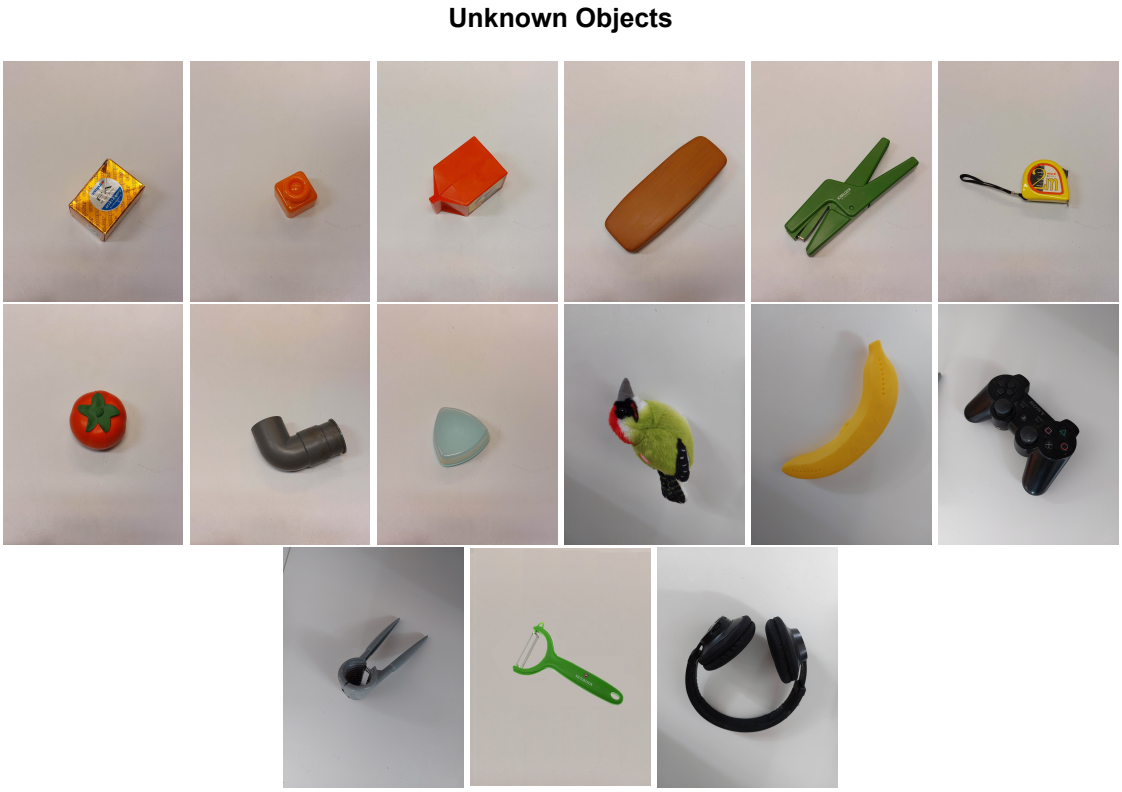


Figure B.2. Unknown objects during the experiments.

C

Network predictions

This appendix contains a selection of the network predictions of experiment 1, experiment 2 and experiment 3.

Figure C.1. Grasp predictions for images from experiment 1. The first and second column are the RGB images. The second and fourth column contain the predictions. The predicted pixels of the objects are covered yellow, the predicted grasps are given in green and the grasp that is executed is shown in blue.

Figure C.2. A sequence of grasps of experiment 2. The first and third row contain the RGB image at the grasp area. The second and fourth row gives the predictions by the grasp network. The predicted pixels of the objects are covered yellow, the predicted grasps are given in green and the grasp that is executed is shown in blue. Sequence: Rubber duck, Piggy Bank, Wooden Tree, Measuring Tape, Wooden dear and a Triangle box.

Figure C.3. Network predictions from experiment 3. The first and third row contain the RGB-image at the grasp area. The second and fourth row gives the predictions by the grasp network. The predicted pixels of the objects are covered yellow, the predicted grasps are given in green and the grasp that is executed is shown in blue.