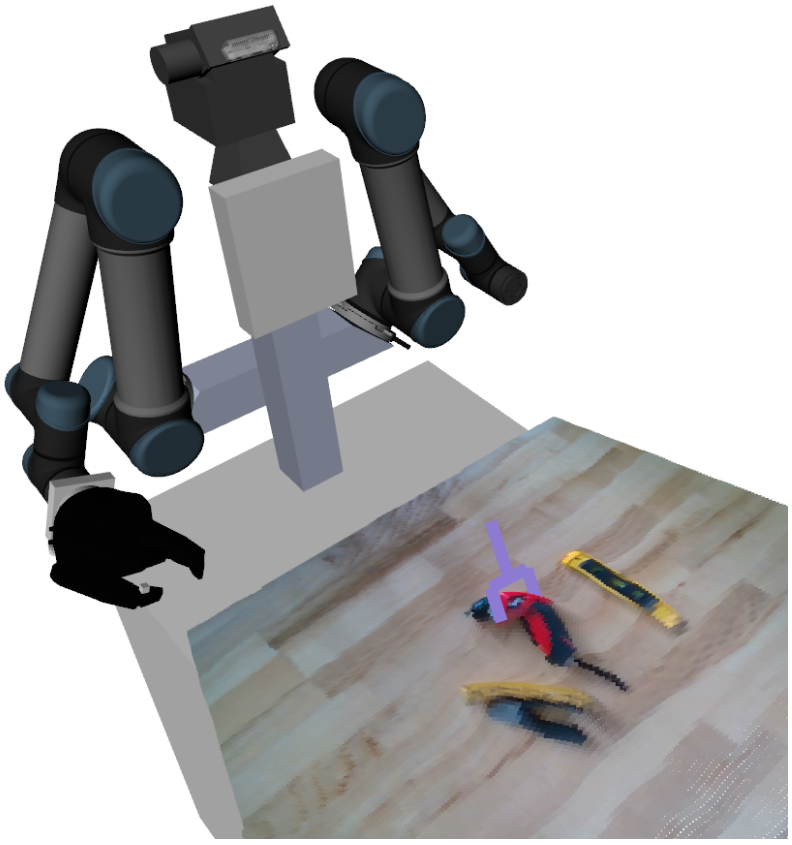
Human-to-Robot Handovers Based on Visual Data for Optimisation of Industrial Tasks

Handovers and Grasp Generation

Master Thesis

Group 1061



AALBORG UNIVERSITY

DENMARK

Department of Electronic Systems

Attributions

This report was typeset using LaTeX.

AALBORG UNIVERSITY
DENMARK

Department of Electronic Systems
Department of Materials and Production
M.Sc. Robotics

Fredrik Bajers Vej 7
9220 Aalborg Ø
http://www.es.aau.dk

Title:

Human-to-Robot Handovers Based on Visual Data for Optimisation of Industrial Tasks

Theme:

Master Thesis

Project Period:

Spring 2021

Project Group:

Group **1061**

Authors:

Jan Kjær Jørgensen
Rune Grønhøj

Supervisors:

Dimitris Chrysostomou

Number of Pages: 48
Appendix: 0 pages
Date of completion:
June 2, 2021

Abstract:

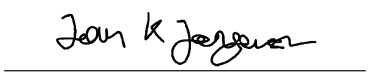
This project revolves around human-to-robot handovers, with a focus on robust real-time grasping. The system is developed with the Little Helper 7 dual-arm UR5 platform in mind. This project explores multiple state of the art grasp generation methods to utilise them for real-time grasping in handover scenarios. A standardised conversion between grasp representation is presented for visualisation of grasp predictions using ROS. Multiple viewpoints are investigated using a custom grasp rectangle data-set, with 50 scenes from three different views. Two pixel-wise real-time grasp generation methods (GG-CNN and GR-ConvNet) are explored and tuned, including tweaking batch-size, optimiser and data-set. Mainly, the novel Graspnet 1-billion data-set is investigated to improve the existing performance of the models, as the data-set is widely different from the previously available ones. The grasp generation methods proposed shows promise during evaluation. Especially the use of Graspnet seems to improve invariance to viewpoints which is essential during a handover scenario. However, to achieve a human to robot handovers, integration with other systems is needed. Mainly hand detection is required and integration with previous systems of the Little Helper 7.

Preface

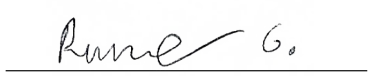
The report *"Human-to-Robot Handovers Based on Visual Data for Optimisation of Industrial Tasks"* are written by group 1061 as our thesis for the Robotics Master program at Aalborg University.

Firstly, we would like to thank our supervisor, Dimitris Chrysostomou, for motivation, good talks and support throughout both the bachelor and masters education. He always supplied the desired hardware (and more), listened to our weird ideas and helped shape our project proposals and education in general.

Furthermore, we also want to express thanks to our friends and families for continued support throughout the last 5 years of our journey through university.



Jan Kjær Jørgensen
<jkja16@student.aau.dk>



Rune Grønhøj
<rgranh16@student.aau.dk>

Reading Guide

A Git repository for the project is made available on Bitbucket with the following link:

`https://bitbucket.org/masterrob/lh7-handover/src/main/`

The list of acronyms ordered alphabetically

Acronym	**Definition**
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DOF	Degrees of Freedom
FPS	Frames per Second
GPU	Graphics Processing Unit
HRC	Human Robot Collaboration
HRI	Human Robot Interaction
IOU	Intersection over Union
LH7	Little Helper 7
LIDAR	Light Detection and Ranging
RGB-D	Red, Green, Blue - Depth (Images)
ROS	Robotic Operating System
UR	Universal Robots
YOLO	You Only Look Once

Table of Contents

1 Introduction

This chapter introduces the overall topic revolving around the theme of the master thesis: Human-robot handovers. An overview of the hardware and software components of the robotic system is given. It recaps the development of the Little Helper 7 platform and examines previously developed functionalities as stepping stones for new development. Finally, the project is given a specific direction, which is the basis for the contributions of the project. Finally, an outline of the thesis is presented.

As artificial intelligence for robotics is on the rise, the usability of robots and relation and collaboration with human increases and becomes more advantageous[1]. Human-robot collaboration (HRC) revolves around synergistically combining humans and robots to utilise the individual strengths of both the humans and the robots[2]. A major aspect of HRC is the task of collaborating with a robot to achieve a common goal. One of the main tasks that enable this teamwork is handovers. Humans do handovers with little to no thought, as it is an inherently learned ability[1]. However, this is a complex task for robots to accomplish, touching on topics such as:

- **Communication** (agreeing on time and place of handover, often non-verbally)
- **Motion planning** (moving safely and predictably)
- **Safety** (not hitting or harming the human)
- **Grasping** (taking hold of an object without dropping it)

With these main essential elements for human to robot handovers in mind, the robotic system used for this project is explored. Some of the aspects are covered by previous development of the robotic platform, and therefore the scope of this project is focused on specific areas.

1.1 Previous Development

In this project, the Little Helper 7 dual-arm robotic platform is utilized (see Figure 1.1). It contains two UR5 manipulators, a Robotiq 3-finger gripper, a camera on the wrist of the arm with the gripper, a camera mounted on a pan-tilt unit at the top of the system, and a screen in the middle of the torso, in the form of an iPad. Furthermore, a LIDAR is positioned in front of the robot. In addition to this, at the bottom, a base is found, consisting of a move-able table and a compartment for the controllers of the manipulators and the central computer that controls all the systems.

The Little Helper robotic family is a long-running project, where Little Helper 7 is the seventh iteration. This iteration mainly differs from previous versions with its dual-arm

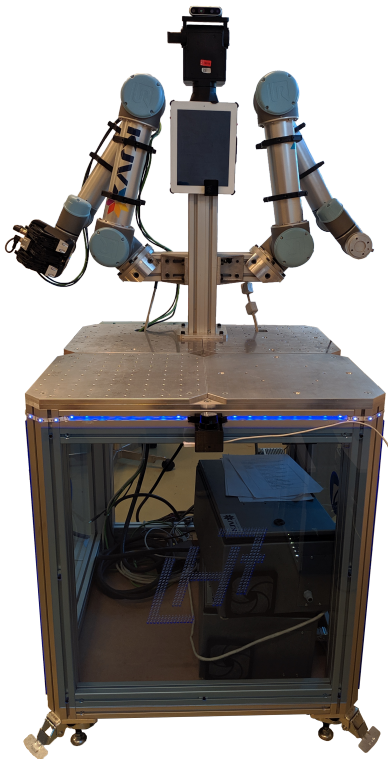


Figure 1.1: The physical Little Helper 7 platform.[3]

capabilities. The system has been in development for three years, starting with the construction of the platform [4]. With the platform as a base, functionalities have been incrementally added throughout semester projects.

Firstly, an industrial disassembly task was considered, where a mock-up product was dynamically disassembled using a vision system [5]. Furthermore, development was made in [6] where a task-oriented dialogue system was explored to enable natural language communication during pick and place tasks, where also a vision system is utilized for object detection and grasp generation - however, only realized in a simulated environment. Finally, HRI/HRC aspects are explored, where the human around the robot is detected and tracked for safety measures while also recognising the performed actions [3].

This work focuses on human-to-robot handover. This includes object manipulation in the form of object-independent grasp generation. In combination with the following existing sub-system it will be possible to realise a full handover scenario:

- **Action recognition** (Initiation of handover)
- **Human pose estimation and tracking** (Safety and location of human)
- **Motion planning** (Robot movement)

These systems allow the realization of a human to robot handover. However, one important system is missing, that being object manipulation, or more specifically *grasp generation*. Figure 1.2 shows the flow from the hardware and software systems. The camera and LIDAR input is shown at the top, where they first are used for human awareness in the form of human pose estimation, tracking and action recognition. These elements are essential for

handovers as they help ensure safety and can be used to initiate the handover. Following human awareness is object manipulation, which is a missing part of the Little Helper 7 system and the focus of this work. This system gets input from the RGB-D camera for grasp generation. Finally, a method for control of the robot exists, which is concerned with planning the movement of the manipulators and moving them in the desired manner.

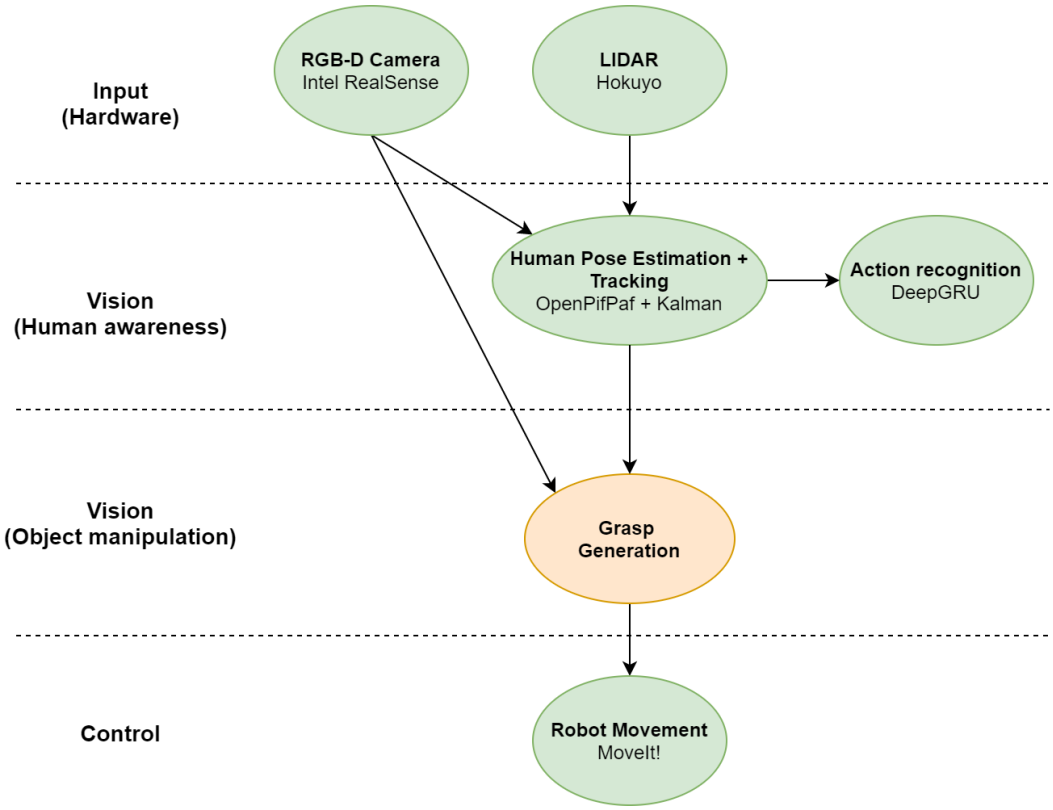


Figure 1.2: System diagram showing in green the previously implemented systems and in orange the areas of focus for this project.

Grasp generation is chosen as the focus of the project, as it is the most crucial missing system needed to realise a human to robot handover action. Furthermore, it has an essential role in handovers and therefore needs to be explored extensively to achieve successful human-robot interaction.

1.2 Contributions

Given the previously developed systems and functionalities of the Little Helper 7 platform, and the focus area of grasp generation for this work, the project has the following contributions:

- Exploration of different state of the art grasp generation methods
- Standardized conversion between different grasps for visualisation
- Fine-tuning of a set of robust real-time grasp generation methods
- A new viewpoint data-set for grasp generation evaluation
- Extensive evaluation of a selection of grasp generation methods
- Proposal of necessary future work to bring efficient grasping into handover actions and thereafter to operations in the industry

1.3 Outline

Chapter 1 introduced the project. It also gives an overview of the robotic system, the development of the system is recapped, and the handover task is split into smaller tasks, and a specific direction for the project is chosen.

Chapter 2 introduces the concept of grasp generations and gives an overview of the significant developments of grasp generation, the input modalities, grasp representation and visualisation. Furthermore, in Chapter 3, grasping data-sets are examined to understand the data which is needed to train a network for grasp generation.

The details and methods used for grasp generation are presented in Chapter 4. This includes the choice of grasp generation network, among the ones explored in Chapter 2. Furthermore, details on training, data processing and data loading are covered.

Chapter 5 presents the evaluation of the chosen methods. Herein, the testing performed to evaluate the general performance. Furthermore, a viewpoint data-set is created to understand the invariance of the models to changes in viewpoint.

In Chapter 6, a discussion about the main elements of the work is comprised. It also includes a discussion on the performed evaluation. The concept of human to robot handovers is also considered again and set into perspective with the development of the grasp generation method. Finally, in Chapter 7, a conclusion is given that brings the project to a close.

2 Grasp Generation for Human to Robot Handovers

When generation grasps, the common goal is, given an input, to calculate the best-known grasp(s). This chapter explores important works in grasp generation literature. Grasp generation exists in many cases and is widely differed for different gripper types, like: Parallel jaws, soft grippers or fully articulated grippers resembling a human hand. However, for this project, the focus lies on antipodal grasping methods using parallel jaw grippers as it achieves good results despite its simplicity and is widely studied and researched.

Antipodal grasping is a versatile method of manipulating an object as many grippers are parallel inherently or can perform a parallel action. In this project, the Robotiq 3-finger gripper is used in pinch mode, imitating a parallel gripping motion.

Furthermore, only newer advances in the field concerned with neural networks and deep learning are considered. Within this scope, image, depth and point cloud modalities exist and is explored in this chapter. Furthermore, multiple different grasp representations are investigated in Section 2.3.

2.1 Related Works

Firstly, this section covers related works in the domain of handovers. Furthermore, it delves into robotic grasp generation, which can be used as a significant part of the handover action. The grasp generation methods are all based on antipodal grasping.

2.1.1 Handovers in Literature

The act of handing over an object is an essential physical interaction skill during daily living. Oftentimes the need to transfer an object to another person arises, whether at home, at a restaurant or in an industrial workplace. Therefore, the handover skill is essential for assistant robots in many different areas, as this physical interaction is useful in many scenarios.[7]

The handover usually involves two actors, the *giver* and the *receiver*, that collaborate in successfully transferring an object [7; 1]. The actors need to agree on *when*, *where* and *how* to perform the handover[1].

The *when* of the handover concerns the timing aspects of the collaboration, not only how the handover is started, the initiation, but also the timing of when the *giver* should let go

of the object to allow the *receiver* take it. An important aspect is gaze. Studies [8; 9] show that gaze can improve the handover efficiency and feel, as it is a simple form of underlying communication that indicates the intention of both actors in the handover. Furthermore, initiation concepts where human motion or gestures is used is shown in [10; 11], where the motion of the human body or hand is tracked and categorized. The detected action can then be used to alter the behaviour of the robot, e.g. initiating a handover, when the arm is moved towards the robot or the hand is opened, ready to receive an object.

where is a bit more simple, as it determines the location in the space where the handover is to take place. This location is usually in the midpoint between the two actors or shifted towards the *receiver*[12].

The process of *how* can be split into two main parts: The movement of the human/robot and the grasping technique used to get a hold of the object.

In the past, robot safety was established by physically separating the human and the robot, by cages or by detecting the person entering the workspace using sensors[13]; however, when performing handover, there is a need for the human and robot to be close, and safety becomes a different and important task. The movement of the robot arm is therefore important, as it has to be safe for the collaboration to succeed[14]. Furthermore, to achieve a successful handover for the human collaborating with the robot, it has to move in predictable ways, as it enables the human to understand the robot, which makes the handover more efficient and comfortable[11; 15].

Finally, grasping is needed to manipulate the objects. In [16] the grasp is pre-defined based on the small selection of objects, and only the approach is explored. However, to achieve a dynamic system capable of manipulating a majority of objects, grasp generation is needed[14; 17].

Therefore, grasping literature is explored for methods that enable robust real-time grasp detection, as a means to achieve handovers.

2.1.2 Grasping in Literature

The introduction of neural networks and deep learning for grasp generation started in [18], where RGB-D images are used in a sliding windows-like manner to generate rectangle grasps. This sparked the deep learning research in the grasping domain. In [19] a single look approach is introduced, similar to YOLO[1][20], which drastically improved performance and made grasp generation viable for robotic manipulation in real-time.

In more recent work, the 6-DOF grasp representation is explored. While image-space representations have inherent computational advantages, generation grasps in space have clear advantages in manipulation of real-world object[21]. The first major introduction to 6-DOF grasp generation [22] uses point cloud as input and is widely used as a baseline for these types of methods. Although a new closer to real robot grasping 3D approach is presented, it still is not deemed reliable enough to be viable[22].

In [23] a new 2D approach is introduced. Instead of locating rectangles in the image by

[1]State of the art object detector

looking at objects or the scene in the previously utilized ways, the network predicts three images, where each pixel of the images corresponds to a parameter of the grasp. The three images represent: Gripper width, grasp angle and grasp quality (the grasp is implied to be centred around each pixel, eliminating the need to specify grasp centre separately). An example can be seen in Figure 2.1. This work inspired other works in 2D [24; 25] and even similar works in 3D [26].

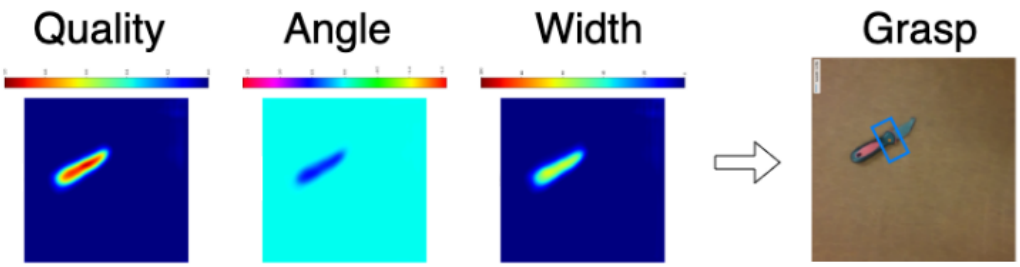


Figure 2.1: Example of three images representing a grasp at each pixel[24].

All the methods using this pixel-wise grasp technique have vastly improved the inference speed of grasp generation. Furthermore, 100's of grasp candidates can be found at each frame due to the nature of predicting a grasp at every pixel. This also easily makes it possible to scale the method to more grasp predictions per image, which is especially useful in cluttered scenes.

A different approach is introduced with Dex-net [27], which uses sample-based grasp candidates and a grasp quality (GQ-CNN) network to pick the best grasp from randomly sampled grasps on an object. GQ-CNN was training on a large grasping data-set of 6.8 million grasps created from synthetically captured point clouds of 1500 3D household object models in a simulation. Each grasp in the data-set is evaluated with the robust grasp wrench space, which analyses a set of contact points from a grasp on an object, and tests if the grasp can withstand external wrenches. These wrenches include gravity, friction and the collision with the table and other objects. The grasp is a force-closure grasp if all the external wrenches can be compensated for during the grasp.[28; 27]

As technologies behind hardware become better due to new developments, more complex grasp generation methods arise, which can uphold the same inference time, despite more complex computations. In [21], a partial point cloud is observed, alongside proposing grasps, the point cloud is reconstructed and used to refine the grasp. Reconstruction of the partial point cloud requires more computational power; however, it also gives a more detailed object representation and hence grasp.

Furthermore, different representations of the 3D space are explored for learning in [26]. Here a point cloud is captured and used to create a TSDF[2], which is then used for training a network for detecting 6-DOF grasps. Promising results are shown, and the idea of representing the data in different ways is introduced.

Finally, in [29] a grasping data-set is introduced. It is created to close a gap in research, where there is not sufficient data available for training grasp generation methods for

[2]Truncated Signed Distance Function

cluttered scenes. In addition to the vast data-set, a baseline prediction method is proposed, which predicts 6-DOF grasps in cluttered scenes at varying viewpoints.

An overview of the previously mentioned methods and other similar ones are compiled in Table 2.1. It contains essential information about the different methods, including: The backbone network or algorithm used for grasp generation; The main data-set which is used to train the model (see also Chapter 3); The input modality; Whether the predicted grasp is presented on an image or in space; Information about the scenes which it was trained on; and finally the inference time, if given in the paper.

The input modality, the viewpoints and the grasp representations are presented following the table.

Ref.	Network	Dataset	Input	Out.	View	Cam. location	FPS
[18]	DeepNet	Cornell	RGB-D	2D	Single object	Head	0.07
[19]	AlexNet*	Cornell	RG-D	2D	Single object	Top	50.00
[30]	ZF (Conv)	CMU	RGB	2D	Clutter	Top	
[31]	LeNet	Custom (BigBird)	Pointcloud	3D	Clutter	Wrist+Side(2)	
[32]	CNN	Custom (Fruits)	RGB-D	2D	Clutter	Top	100.00
[33]	DCNN	Cornell + Washington	RGB-D	2D	Clutter	Head/Top	7.10
[34]	AlexNet	Custom	RGB	2D	Clutter	Head	
[22]	LeNet	Custom (BigBird)	Pointcloud	3D	Clutter	Wrist+Side(2)	
[35]	DGGN	Custom	RGB-D	3D	Single object	Head/Side	
[23]	GG-CNN	Cornell, Jacquard	Depth image	2D	Clutter	Wrist	166.67
[36]	ResNet-50*	Cornell, grasp_multiObject	RGB-D	2D	Clutter	Head	8.33
[37]	ResNet-101	Custom (MIT)	RGB-D	3D	Clutter	Multiple	16.67
[38]	PointNet++*	Custom (ShapeNet)	Pointcloud	3D	Single object	Wrist	
[24]	GR-ConvNet	Cornell, Jacquard	n-channel images	2D	Clutter	Top	50.00
[25]	Fully Conv	Cornell	RGB-D	2D	Single object	Top	125.00
[27]	GQ-CNN	Dexnet 4.0	Depth image	3D	Clutter	Top	
[39]	PointNet*	YCB Dataset	Pointcloud	3D	Clutter	Wrist	
[40]	ResNet/VGG16 + RPN	Custom (VMRD), Cornell, Jacquard	RGB	2D	Clutter	Head	9.10
[41]	PointNet++*	Custom	RGB-D	3D	Clutter	Wrist	0.40
[29]	PointNet++	GraspNet	RGB-D/Pointcloud	3D	Clutter	Wrist	
[42]	3D CNN	Custom (simple shapes)	Pointcloud	3D	Clutter	Side	
[43]	DNN + REM	Cornell	RG-D	2D	Clutter	Top	50.00
[21]	Res-Net-34*	Custom (YCB, BigBird, "Shoes")	RGB-D	3D	Single object	Side/Head	-
[44]	PointNet++	Custom (YCB)	Pointcloud	3D	Clutter	Wrist	79.37
[45]	R-FCN + Angle Net	Cornell	RGB-D	2D	Single object	Top	17.50
[26]	VGN	Custom Synthethic	3D voxel grid	3D	Clutter	Wrist	100.00

*Table 2.1: Overview of deep learning grasp generations methods explored for the project. *=modified*

2.2 Input Modalities

Different methods use different input modalities. The two main categories are 2D and 3D data. The 2D data consists of RGB or grey-scale images; however, in most works, 3D input data is used. The simplest form is RGB-D data, where also a depth image is used, although sometimes the data is structured in a point cloud. RGB-D and organized point cloud data are, given the camera parameters, equivalent[46]. As point cloud data can be generated from aligned RGB-D images.

In some cases, the blue channel of the image is replaced with the depth information, as it makes it possible to use existing networks designed for 3-channel images, with minor loss[19; 43].

The choice of input modality is important as it determines how the data has to be captured. Furthermore, it also defines the level of detail that can be expected in the output. When using RGB data solely, a more flat representation of the world is used, which in turn also impacts the level of detail that can be captured in a grasp. However, when using point clouds, much more detail is known about the scene, and more intricate grasps can be generated. Using 2D data is inherently faster than point clouds due to the amount of data and the level of detail captured. Using RGB-D data is a middle ground where depth information is included in a somewhat flat data structure, which makes it a good balance between precision and speed.

2.2.1 Camera Viewpoint

Another important factor to consider is the viewpoint from which the images are taken. Four main viewpoints are considered: *Top*, *Head*, *Side*, and *Wrist*. The first three can be seen in Figure 2.2. *Wrist* is left out since it is a viewpoint where the camera is mounted to the wrist of a robotic manipulator, and hence the viewpoint can change and be considered a combination of the former three.

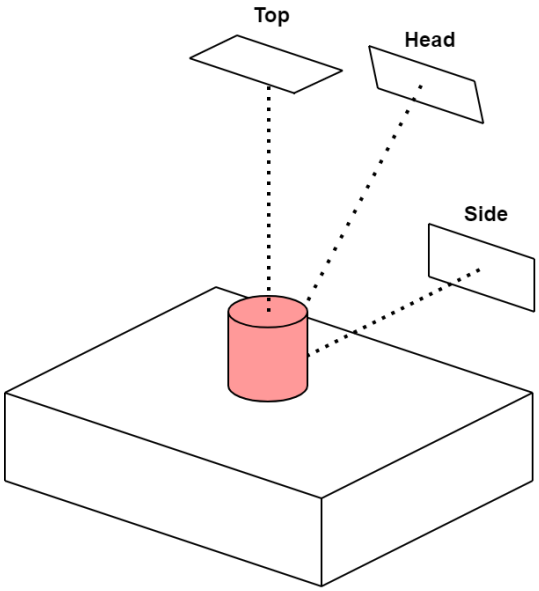


Figure 2.2: Different camera viewing angles, when observing an object (red cylinder) to be grasped.

Some methods work entirely on pre-captured images and do not consider a robotic system, and hence solely try to optimize performance on a set of data. However, the images in the data-sets have usually been taken in one of the four configurations. The different viewpoints used in the referred literature for the report can be seen in Table 2.1.

When designing a system and deciding on a data-set, it is vital to consider the viewpoint. Having data from a single viewpoint will make it harder for a system to generalize to other viewpoints if new data is presented like that. It is suggested that the training data somehow reflects the viewpoint that is used during inference.

2.3 Grasp Representation

For a machine or a robot to understand and use the notion of a grasp, it needs to represented in a specific way. This representation is used to understand where and how to grasp an object, based on a set of numbers. Different methods exists and are explored in this section. Mainly two different methods exist, a planar (2D) method which can be used directly on images and usually assume that the object is grasped from the top. And a 6-DOF (3D) methods, which describes both the location and rotation of the gripper in space.

For 2D grasp representation, the rectangle grasp methods is usually used. It was introduced in [47] along with the Cornell data-set. It consists of the point of the upper left corner of the rectangle, a width and height and an angle. This representation was modified in [19] and became the new standard for planer grasp representations, see Figure 2.3. The representation includes the point at the center of the grasp, the height and width and an angle, it is usually presented as:

$$g = \{x, y, \theta, h, w\} \tag{2.1}$$

Figure 2.3: Rectangle grasp representation. ***Left****: Introduction of the representation[47].* ***Right****: Modified rectangle representation[19].*

This representation is widely used, however, sometimes modified to not include the height or additionally include a quality[3] measure of the grasp [36; 37; 23; 40; 24; 43].

It is also important to note, that the exact way that the rectangles are presented in the raw data differs between data-sets and methods. Some methods define the coordinates of the four corners of the rectangle, while other base the data around the center of the rectangle. However, they can all be converted to the format in equation 2.3.

[3] Usually denoted q or z

The first mayor 3D grasp detection methods arose in [22; 31], and hence the 6-DOF grasp representation, as shown in Figure 2.4. Usually it consists of two elements: A rotation and a translation. The translation is always specified as a vector of (x, y, z) which determines the location in space of the grasp. Furthermore, the rotation gives the orientation of the grasp/gripper, however, the representation of the rotation differs between Euler angles (r_x, r_y, r_z) [39; 42], quaternions (x, y, z, w) [26] or 3x3 rotation matrices [16; 41; 38], as there exist no general standard for 6-DOF grasps.

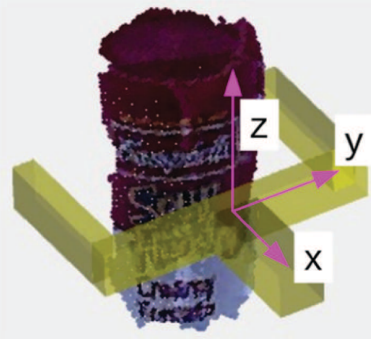


Figure 2.4: 6-DOF grasp representation as shown in [22].

Furthermore, the rotation and location of the coordinate system of the grasp (see Figure 2.4), also does not have a standard, and hence differs between the methods. Typically, it is placed as on the figure, however, the z-axis is often pointing the same direction as the gripper fingers. Some methods also center the grasp between the gripper fingers or at the object.

2.3.1 Converting Between Representations

For comparing different grasp generation methods, conversion between representations is necessary. This is also important when dealing with different data-sets as they are not standardized, and hence have different ways of representing a bounding box. Therefore, a common format have to be chosen when loading the data, which will be used when loading data from other data-sets. It can either be specifying the box by the four corners of it, or basing it around the center and the distance to the edges and an angle. Luckily, the conversion is easily performed using trigonometry.

In addition, conversion from a planar representation to a grasp in space is needed when a physical robot needs to actually perform the grasp.

To convert from the 2D representation, firstly both the representation and a notion of the distance to the object (depth) is needed. This comes naturally for methods that use RGB-D data as input data. The center point of the grasp can then be deprojected using the the intrinsic information of the camera used, to get a point in space. When approaching from the top, the angle can be directly used to rotate the gripper and the grasp can then be performed.[48]

When trying to fairly compare a 6-DOF presentation with a planer presentation the opposite conversion can be made. However, some data will be lost and in most practical examples this is not needed or used. However, the point can be projected to the image

with the intrinsic information of the camera. Then it can be assumed that the object is grasped from the top (like with planar grasping) and the rotation of the approach vector can be used for specifying the angle θ.

These conversions can also be used for visualising the grasps in the same manner, whether that be on an image or in space. This is useful for qualitative evaluation and comparison of different methods.

2.3.2 Visualisation

These conversion methods can be used to make a visualiser that given different grasp representation can visualise 2D and 3D grasps, for comparison of methods. A general grasp structure is created, which holds the parameters of both a rectangle and a 6-DOF grasp. The incoming grasp is then converted, so that both representations are available and filled into the standardized grasp structure. The general pipeline is shown in Figure 2.5.

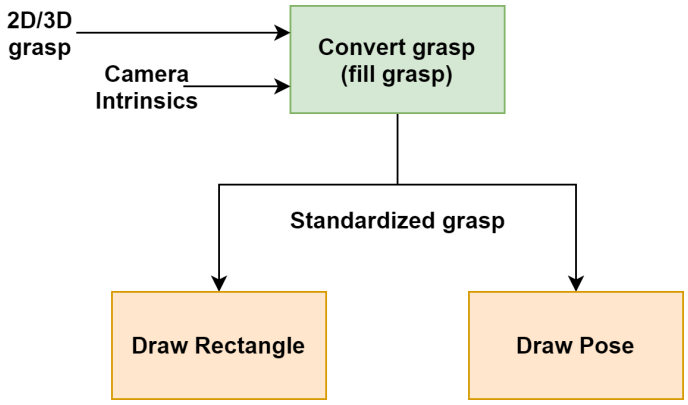


Figure 2.5: Outline of the process of the grasps for visualisation.

An example based on Figure 2.5 could be: A grasp generation methods predicts a 2D rectangle grasp. It is then converted to a 3D 6-DOF grasp using the intrinsic parameters of the camera. Both the representations are then combined in the same data structure that contains all the information, which can then be send to different parts of the system, like the visualizer.

The visualizer is implemented using ROS, and visualisation is done in Rviz. The rectangles are draw on the image from the camera using openCV and published as a ROS Image, similarly the 6-DOF grasp is created as a pose and a simple gripper object is drawn using markers, to represent the grasp. These markers are also published to be visualised in Rviz.

An example of the visualiser in Rviz can be seen in Figure 2.6. On the left the image from the camera is shown with a grasp rectangle drawn. The center and gripper finger positions are marked with blue circles and the rectangle is drawn in red. In the top left of the image, the quality of the grasp is given, if known from the predictor. On the right the 3D environment is shown, with the robot, the raw point cloud from the camera and a simple gripper marker at the predicted grasp pose.

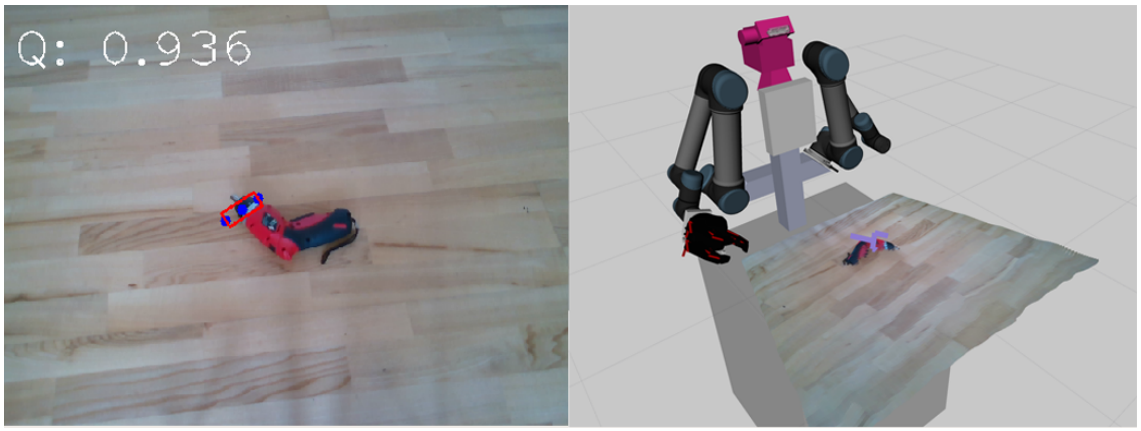


Figure 2.6: Visual representation of 2D and 3D grasps in Rviz. ***Left:*** *RGB image with grasp rectangle.* ***Right:*** *3D grasp in purple shown on point cloud along with LH7.*

2.4 Summary

This chapter presented related works in the handovers domain and grasping literature, with an overview of many state of the art methods, and some methods that shaped the grasping research.

Furthermore, the input modalities used for grasp detections are explored, and also different viewpoints which are commonly used for capturing visual data when using a robot.

Finally, different grasp representations are presented, with mainly 2D grasping rectangle and 6-DOF grasp representations. All of this leading to a grasp visualiser that given any of these two representations can visualise the grasp on both an image and in space.

3 Datasets and Evaluation Metrics

As simple as it is for humans to pick up and grasp everyday objects, it is complex for robotic systems to grasp objects. In recent years, using deep neural networks for predicting grasps from RGB-D image and point clouds has emerged[49]. These networks rely on large amounts of data for training, using supervised learning[50]. Therefore labelled data is needed, which can be generated by hand with human labelling or automatically with algorithms for labelling data. Recently, the creation of large and diverse data-sets have emerged in research. The data-sets consists of scenes containing one object or cluttered scenes with multiple objects[29; 51]. These scenes are commonly labelled using the rectangle grasp representation as described in Section 2.3, as well as some newer adaptations of 6-DOF grasp labelled data.

For robotic grasping, data-sets are used as a standardization, making it easier for other researchers to reproduce the results, as it gives access to the same objects, labels and data in general. Some robotic grasping methods purely train on synthetic data generated in simulation, while others generate a data-set from real-world objects.

This chapter explores the major data-sets used for training and evaluating deep neural networks for grasp generation. This includes RGB-D and point cloud data and object data-sets, which are commonly used to generate synthetic data. Furthermore, only object and grasping data-sets commonly used in research and publicly available are considered.

3.1 Object Data-sets

Large objects-sets have been shown to be vital [50] for benchmark and generating training data for grasping methods in the real world and simulation. In recent years, researchers adopted random objects from the office or the use of a subset of objects from larger object data-sets like Shapenet [52] or Bigbird, [53] which are graspable to use for training and benchmark.

The Columbia grasping data-set [54] was one of the first to introduce a 3D object set in GraspIt[1] simulation for generating grasp labels with over 8000 grasps of everyday objects.

Bigbird and YCB [53; 55] introduced two databases of high-quality 3D scans of 75-125 real everyday objects, which can be used for generating synthetic training data in simulation for robotic grasping or object detection algorithms.

Shapenet [52] provides researchers with a huge database of 3D shapes and objects spanning

[1] A grasping simulation software

over 55 common[2] categories. Commonly researchers choose a subset of objects in a few categories which contain graspable everyday objects, e.g. bowls, cans, tools, fruit, etc.[56; 50]

The ACRV Picking Benchmark (APB) [57] created a robotic benchmark challenge for driving the scientific progress in robotic grasping. The benchmark is designed for reproducibility and consist of 42 well-defined household objects and methods for evaluating the researcher's systems. Each object is well-defined as it has a high-density 3D model. Furthermore, the objects are described in details so that they can be bought physically. Finally, the placement of the objects in the world for the benchmark is specified for reproduction.

Evolved Grasping Analysis Data-set (EGAD) [50] introduces a new robotic grasping data-set aimed at generating objects which vary in shape complexity and grasp difficulty, as seen in Figure 3.1. EGAD tries to standardize robotic grasping data-sets by making 3D object shapes that span the whole spectrum from easy to difficult in grasping objects and shape complexity. Making the data-set 3D printable makes it easier for researchers to test their systems instead of buying the physical objects and standardising the objects for other researchers to reproduce results.

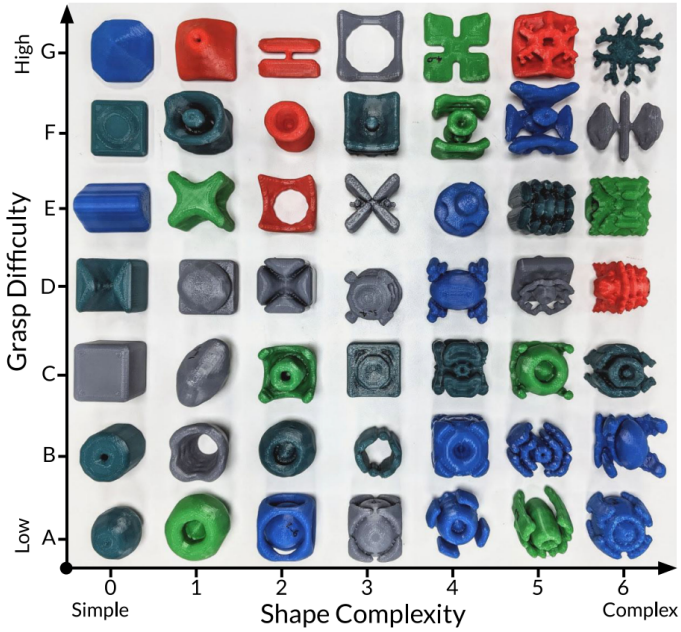


Figure 3.1: A subset of the 3D generated objects from the data-set, increasing in shape complexity and grasp difficulty [50].

Table 3.1 shows a list of the commonly used 3D object data-sets explained above that are used for the generation of grasping data for training and benchmarking grasping methods. The table contains the amount and type of the objects for each data-set. It also includes the data type of the data and whether it includes grasp annotations.

[2] vehicles, furniture, household etc.

Object Dataset	Objects	Description	Data type	Grasps
Columbia [54; 58]	8000	Household objects	Meshes (.IV, .XML)	Yes
Bigbird [53; 59]	125	Household objects	RGB-D (.jpg, .h5)	No
YCB [55; 60]	75	Household objects	Meshes (.URDF, .XML)	No
Shapenet [52; 61]	51300	55 Categories of objects	CAD (.stl)	No
Dex-Net 2.0 [56; 62]	1500	Complex objects	Object (.HDF5)	Yes
APB [57; 63]	42	Household objects	-	Yes
EGAD [50; 64]	2282	Simple to complex objects	Meshes (.obj)	Yes

Table 3.1: Commonly used object data-sets for generating training and testing data. The table contains references to the paper of the data-sets as well as the link to download the data-set. List is ordered chronologically.

Table 3.1 shows that household objects are the most common object type used for generating grasping annotation. Only Dex-Net and EGAD deviates from this and use a mix of simple and complex shapes. Bigbird, YCB and APB contain fewer objects compared to the other. It is speculated that this is due to the data being high-quality scans of real objects, which cost both time and resources to create. Furthermore, they are usually only used for benchmarks, which might also explain the smaller amount of objects in general.

3.2 Grasping Data-sets

Using the object data-sets mentioned above, researchers created grasping data-sets for training deep neural networks. However, some researchers create grasping data-set independently from the object data-sets. They all revolve around using a set of objects to create scenes of objects, with ground truth bounding box annotations for possible grasps. The number of objects and grasps per scene vary among the different data-sets. Using the object data-sets mentioned above, researchers created grasping data-sets for training deep neural networks. However, some researchers create grasping data-set independently from the object data-sets. They all revolve around using a set of objects to create scenes of objects, with ground truth bounding box annotations for possible grasps. The number of objects and grasps per scene vary among the different data-sets.

This section explores commonly used grasping data-sets among researchers, starting from the Cornell data-set, which sparked many other works until the newly developed Graspnet data-set.

The Cornell data-set[47] introduced a grasping data-set with a single object per image and a new rectangle grasp representation (see Section 2.3) used to label grasps. The data-set has 8019 grasps in RGB-D images, which were manually labelled by hand; some examples are shown in Figure 3.2. The idea of using RGB-D data labelled with rectangle grasps continue in [36; 40] where cluttered scenes are introduced, with multiple objects per scene to achieve better grasp performance in cluttered scenes.

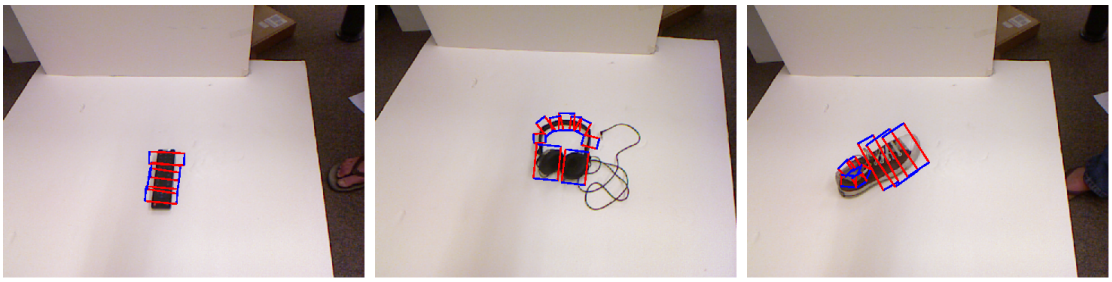


Figure 3.2: Examples from the Cornell data-set. Rectangle grasps drawn on raw image[47].

The previously mentioned data-sets were labelled by hand, which is time-consuming and prone to error. The Jacquard data-set[51] expands on Cornell's rectangle representation by automating grasp generation and labelling in a simulated environment. Grasps candidates are generated across the object and tested in the simulation; if the object could be lifted from the ground, the grasp was considered correct and annotated accordingly. Finally, this results in a large data-set with a total of 1.1 million grasps across 54000 images. An example of grasp annotations from the simulation can be seen in Figure 3.3.

Figure 3.3: Examples from the Jacquard data-set. Rectangle grasps drawn on raw image[51].

Dex-Net 2.0[56] introduces a similar method by simulating the point clouds of 3D objects and generating grasp candidates based on the synthetic point clouds. However, they use their own grasp representation and generate the grasps using grasp wrench space[28]. The grasps calculated are force-closure grasps if all wrenches are compensated for when the gripper lifts the object off the ground.

Other works, like [65; 34] looked into using between 6 and 14 robots for generating a grasping data-set from robotic trials where the robots would move to a position to pick up objects, and if the object was lifted, it was noted as a positive grasp, if not it was noted as a negative grasp. A similar technique is used in [35], where humans grasp objects in virtual reality to generate a grasping data-set, which closely relates to how humans normally would grasp an object.

More recently, [29] introduced the Graspnet 1-billion data-set, which consist of more than 1 billion grasps following the 6 DOF grasp annotation, which can be converted to the rectangle grasp annotation. Graspnet expands on [36; 40] by using cluttered scenes with multiple objects and capturing RGB-D images from these scenes. For each scene, images are captured from two cameras equipped on a robot's end-effector moving to 256 different

positions covering the whole scene while recording the robot's position. Each scene consists of randomly chosen objects placed in clutter at different angles and position. The first image from each scene is manually annotated with grasps. In contrast, the rest of the images are annotated autonomously based on annotation from the first image, where all annotations are translated and rotated based on the robot's translation and rotation. Figure 3.4 shows images from 2 different scenes from Graspnet at varying viewpoints.

(a) Scene 1 from different views.

(b) Scene 2 from different views.

Figure 3.4: Examples of two scenes from the Graspnet data-set[29] shown at different viewpoints.

Table 3.2 Shows a list of commonly used grasping data-sets created from simulation or captured from a camera. It is a collection of the aforementioned data-sets, including more specific information about each of them. This includes: The number of objects, grasps and images; The label type; Whether the data is labelled manually or automatically; and the modality used to capture the data. Each data-set also contains a reference to its paper and where to download the data-set.

Grasp Data-set	**Total Objs.**	**Objs. /scene**	**Total Grasps**	**Grasps /scene**	**Total Imgs.**	**Label**	**Auto Label**	**Modality**
Cornell [47; 66]	240	1	8019	∼8	1035	Rect.	No	1 Cam. RGB-D
Pinto et al. [34]	150	15	∼50K	1	∼50K	Rect	Yes	1 Cam. RGB
Dex-Net2.0 [56; 62]	1500	1	6.7M	1	6.7M	Custom	Yes	Sim. Depth
Vr-Grasping [35; 67]	101	1	4.8M	∼100	10K	6-DOF	Yes	Sim. RGB-D
Jacquard [51; 68]	11K	1	1.1M	∼20	54K	Rect	Yes	Sim. RGB-D
Multi-Object [36; 69]	31	3∼5	∼96K	∼100	96	Rect	No	1 Cam. RGB-D
Levine et al.[65; 70]	1100	20	800K	1	800K	Rect	Yes	1 Cam. RGB-D
MIT-Grasp [37; 71]	61	1∼13	-	-	1837	Custom	No	1 Cam. RGB-D
VMRD [40; 72]	∼100	3∼5	100k	∼20	4683	Rect	No	1 Cam. RGB
Graspnet [29; 73]	88	∼10	∼1.2B	3∼9 M	97K	6-DOF /Rect.	Yes	2 Cams. RGB-D

Table 3.2: Commonly used grasping data-sets for robotic manipulation. The table contains references to the paper of the data-sets as well as the link to download the data-set. List is ordered chronologically. "-" denotes the number is unknown.

Table 3.2 shows that data-sets created from simulation have more grasp labels than data-sets captured of physical objects. This is due to the nature of simulation; it is fast and easy to set up for new objects and labelling. Only Graspnet is an exception to this, which used a clever labelling technique for reaching 1.2 billion grasp labels of RGB-D images on a physical object.

After choosing a data-set for training a grasping method, evaluation metrics are needed to determine if the predictions of the model perform well on the data. Evaluation metrics are presented below.

3.3 Evaluation Metrics

When a network is trained using the data mentioned above, some metrics have to be chosen to evaluate the performance of newly seen examples. Many evaluation metrics exist, most commonly, the rectangle metric is used, or a physical evaluation of the grasp using a robot. Furthermore, the new contender, Graspnet, is also explored.

3.3.1 Rectangle Metric

The main metric used to evaluate the Cornell and the Jacquard data-set is the rectangle metric; it is used in most rectangle grasp generation methods and is explained as follows in [19]:

> *The rectangle metric considers a grasp to be correct if both:*
>
> 1. *The grasp angle is within* $30°$ *of the ground truth grasp.*
> 2. *The Jaccard index of the predicted grasp and the ground truth is greater than 25 percent.*
>
> *Where the Jaccard index is given by:*

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

This is similar to the popular *Intersection Over Union* known from object detection (e.g. YOLO[19]), where the intersection and union of the ground truth bounding box and the predicted bounding box is found, and the ratio between them is calculated.

In addition to the rectangle metric, a common way of splitting the Cornell data-set is also used. This is the concept of image-wise and object-wise split as seen in [18; 19; 24]. The image-wise technique splits the images randomly, and the object-wise technique splits the images into groups containing the same objects (or the same type of objects). When evaluating an image-wise split, the performance of determining the location of the objects is found. In contrast, object-wise determines the performance for detecting grasps on novel objects not previously seen. This method helps give a more nuanced picture of how well the model generalizes to locations and novel object independently.

3.3.2 Graspnet Metric

Graspnet also introduces a new metric for evaluating grasp generation methods.

The method is based around 6-DOF grasps. Firstly, to determine if a single grasp is correct, an online evaluation is used. This removes the need for countless ground truths in continuous space with millions of possible grasps.[29]

Firstly, each predicted grasp is matched with a targeted object. The point cloud between the gripper fingers is checked for the correct object. A binary label describing the success of the grasp can then be found by force-closure metric, at a given friction coefficient μ.[29]

However, a single grasp per image is often not sufficient, especially when dealing with cluttered scenes. Therefore, a *precision@k* metric is introduced, where the top-k ranked grasps, are evaluated giving a precision metric.[29]

With this precision metric, AP_μ can be calculated. That being the average precision at friction μ. It is calculated as the average *Precision@k* for k ranging from 1 to 50, given friction μ. Similarly to MS-COCO[74], AP_μ is given at different μ. Where μ range from $\mu = 0.2$ to $\mu = 1.2$, at $\Delta\mu = 0.2$ intervals.[29]

Finally, before doing any evaluation, a pose-NMS[3] is performed, which eliminated similar grasps, especially ones that are on the same object.[29]

3.3.3 Physical Evaluation

In addition to the theoretical measure based on the labelled ground truth, a physical evaluation can take place.

Here, a robot performs the generated grasp to evaluate the performance in a real-world scenario or simulation. Typically the grasp is executed by making the robot approach the grasp from the specified vector (or above if an approach is not defined), moving to the object and closing the gripper. If the robot then successfully lifts the item without dropping it[23] or delivers it to a designated area/box, the grasp is deemed successful. However, if the object is not grasped by the gripper fingers or dropped during manipulation, it is considered a failed attempt.

3.4 Summary

This chapter presented object and grasp data-sets commonly used within grasping research, starting with an overview of the currently available data-sets and how they are used and generated. Furthermore, metrics on evaluating grasping networks are presented based on the rectangle metric, the Graspnet metric and physical robot trials.

With the knowledge of different data-sets and how to evaluate grasping methods, it is possible to dig deeper into a few methods which show potential for HRI in Chapter 4.

[3] Non-Maximum Suppression

4 Grasping Methods and Implementation

Throughout this chapter, more detail is given on a set of chosen methods. This includes a description of how the input images are used to predict bounding boxes. Furthermore, the pre-processing and loading of the data-sets are explored. Including dealing with vastly different amount of ground truth bounding boxes. In addition, the cloud computing service CLAAUDIA used for training is outlined.

Chapter 2 covers a vast variety of grasping methods, and the span of the project is limited; a subset of methods is chosen for continued work in the project. Namely, *Generative Grasping Convolutional Neural Network* (GG-CNN)[23] and *Generative Residual Convolutional Neural Network* (GR-ConvNet)[24] is chosen. Furthermore, it is also chosen to limit the data-sets to the following most commonly used ones: Cornell and Jacquard. Furthermore additionally, Graspnet is chosen.

The Graspnet data-set is chosen as it has a similar format to Cornell and Jacquard but offers a vastly different set of data. Firstly, it has changing viewpoints. Secondly, it has a multitude more bounding box annotations, and lastly, it is new and therefore, not much research has been performed with it. Furthermore, it is speculated that better performance of the networks can be achieved with this data-set, as it has more data to learn from, which ideally would make it better at generalizing.

After discussing the chosen data-sets, the reasoning for the chosen grasp generation methods can be highlighted. One of the main reasons for choosing GG-CNN and GR-ConvNet is the inference speed. They both utilize the same pixel-wise method, making them fast compared to other methods, as seen in Table 4.1. The table shows the inference speed of 4 of the previously mentioned methods. These methods are the ones that were successfully modified to run on the LH7 hardware setup using ROS.

Fast inference is desired in HRI, as humans move in real-time, and if the robot is slow, it will be tedious to interact and cooperate with it.[14] Furthermore, a fast reaction is needed since the object held by the human, most probably will be moving during the handover as it is almost impossible to hold an object completely still in a stretched arm[16].

	GG-CNN	GR-ConvNet	VGN	Graspnet Baseline
FPS	~ 72	~ 23	~ 1.3	~ 10

Table 4.1: Inference speed on implemented methods using ROS framework.

In addition to the inference speed, using RGB-D images also simplifies the data, as it is in fewer dimensions and is easier to handle and modify. This, in turn, usually makes these methods faster during inference.

On the other hand, the limitation also comes from the limited span of the project. Two similar methods are chosen as they have close to the same structure and are easier to adapt data-loaders, training and evaluation for. This is also important as custom data-loaders are needed for the Graspnet data-set, as it has not been used for these networks before.

However, this increased speed also comes at a cost. The representation in 2D of an object is flat compared to a point cloud. Therefore, the objects are only seen from one viewpoint, making it hard to determine the rotation of the gripper in space.

4.1 Methods Implementation

GG-CNN[23] introduced pixel-wise grasping from images in the year 2018. GR-ConvNet[24] expanded the work of GG-CNN in 2019 by introducing a new network architecture. The pixel-wise grasping backbone is the same between the two methods.

This section explores how the two grasping methods work, including the core concept of how grasps are computed pixel-wise and how grasp rectangles from data-sets are used for learning. Furthermore, the section explores how to tune the networks to reach the best training result.

GG-CNN proposes to use Convolutional Neural Network to solve the problem of computing grasps of unknown objects from depth images perpendicular to the surface. The proposed network predicts the quality of grasps and grasps pose at every pixel from a single depth image as input. GR-ConvNet uses a similar approach, with altered network architecture.

4.1.1 Grasp Representation

The two networks detect grasps from a single channel depth image, and optionally an RGB image I and defines a grasp in the image as $g_I = (s, \theta_I, \omega_I, q)$, where $s = (u, v)$ is the center point of the grasp in image coordinates, θ_I is the rotation of the grasp around the camera's frame, ω_I is the grasp width in image coordinates (pixels), and q is the predicted quality of the grasp. Figure 4.1 shows an example of a grasp in the image space.

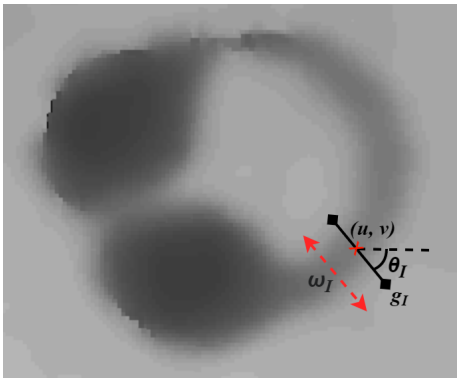


Figure 4.1: Example image of a grasp in the image frame. (u,v) is the center of the grasp in pixels, θ_I is the angle of the grasp and ω_I is the width of the grasp

The network predicts grasps at every pixel from the depth image by defining a grasps map called G in the image space as a set of 3 images (angle, width, quality) denoted as $G = (\Theta, \Omega, Q)$.

These 3 images Θ, Ω and Q contain values from angle θ_I, grasp width ω_I and grasp quality q respectively at every pixel as seen in Figure 4.2. This makes it possible to construct a complete grasp g_I at every pixel, as each pixel of the input image is mapped to pixels on the output images instead of sampling grasp candidates like Dex-net [56].

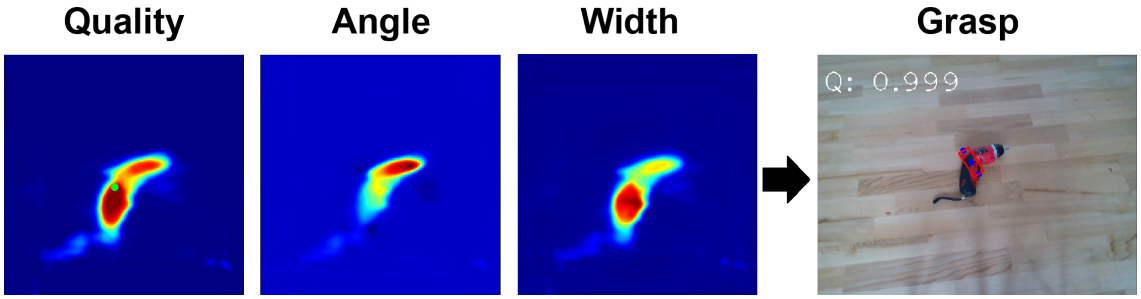


Figure 4.2: Example of the three images quality Q, angle Θ, and width Ω, representing a grasp at each pixel. A colormap is applied to the 3 predicted images for visualisation purposes. The best grasp is drawn as a red rectangle with a blue center dot, on the RGB image.

Q is an image that describes grasp quality with a value between 0-1 at every pixel (u,v); higher numbers indicate a higher chance of a successful grasp. Θ is an image that describes the angle of the grasp with a value from $-\pi/2$ to $\pi/2$ at every pixel. This interval is chosen as it gives a range of 180°, and rotating more would not be necessary since the grasp is symmetric. Lastly, Ω is an image describing the grasping width with a value of 0-150 pixels at every pixel. The width is capped at 150 pixels to ensure that the grasp is contained within the cropped image.

The grasp with the highest probability of success can be found by finding the pixel coordinates (u,v) for the pixel with the highest value in the Q Image. This pixel coordinate can then be used to index the other two images Θ and Ω, to find the angle θ_I, and grasp width ω_I at the same pixel resulting in a full grasp, as seen on the rightmost image on Figure 4.2.

4.1.2 Training the Networks

Training the network consists of using the rectangle bounding boxes from a data-set to construct 3 ground truth images (angle, width, quality) and using the depth image to teach the network to grasp objects.

The rectangle bounding boxes from the data-set consist of 4 corner coordinates describing their shape and location in the image. When loading the data, each ground truth grasp is augmented with the angle of the rectangle relative to the x-axis of the image and with the width of the rectangle (long side of the rectangle).

To train the network, the ground truth rectangle bounding boxes need to be converted to a format, similarly to the desired output, three images. To achieve this, the center third of each rectangle is used to create mask-like images, as seen in Figure 4.3. Only the center third of the rectangle is used when creating the images, as the object to be grasped typically would be in this area, due to the rectangle being drawn a bit away from the objects to leave space for the gripper fingers. The mask-like images are then filled with the values from the ground truth, θ_I and ω_I for angle and width image, respectively and all other pixels are set to 0. Finally, the masked area of the Q-image is set to 1, and all other pixels are set to 0.

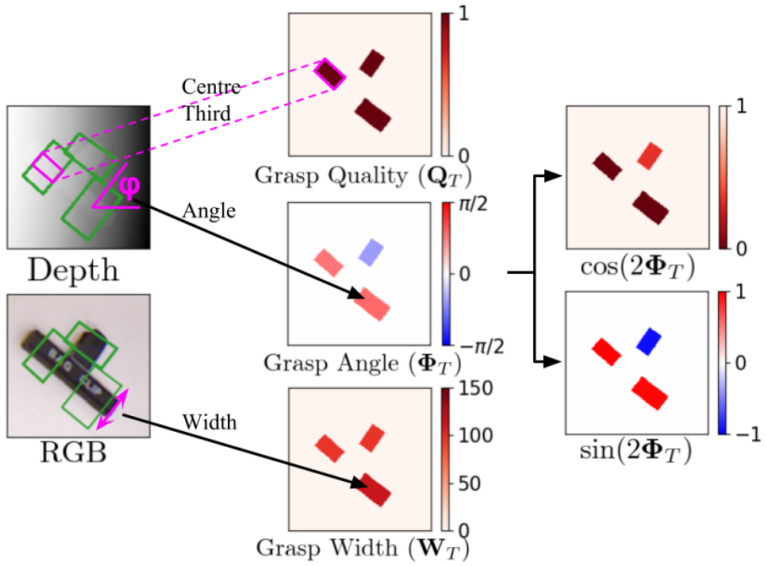


Figure 4.3: Example of ground truth rectangles, being converted to ground truth images. The input depth and RGB image and grasp rectangles are shown on the left. The center third of each rectangle is used to create 3 new images containing grasping quality, angle and width.[23]

As seen in Figure 4.3 the ground truth angle image Θ is converted into 2 images using the trigonometric functions Cosine and Sine, which describes the angle in a value range from -1 to 1. This interval was chosen to make the distribution of values easier for the network to learn.[23]

Once all 4 ground truth images (Q, Θ_{sin}, Θ_{cos}, Ω) are constructed, they are fed to the network with the depth image for training the network. During training random rotation and zooms are applied to the images to increase real-world robustness. The training takes place by computing the mean square error from the ground truth images to the predicted

images. Once one pass of the data-set has been shown to the network, an evaluation takes place. The rectangle metric is used to evaluate the network's performance on the validation set (10% of the data). Finally, the result is used to save the model if the network performs better than the previously best model.

Additionally, to make the images clearer to the human eye, a colourmap is applied to the grayscale images when the images are visualized. Doing this makes it possible to see the peaks more clearly, as grey images are hard to determine the values of with the human eye. For this project, a jet colourmap is used, as it has a wide range of colour in a rainbow-like manner from low valued blue to high valued red.

4.1.3 Tuning Parameters

Different training hyperparameters are tuned when training the model to achieve the best possible performance. For this project, only a few parameters are considered and tuned. The chosen most important parameters are as follows:

- Batch size (Default 8)
- Epochs (Default 50)
- Optimizer (Default Adam)

Each of these parameters is explained in more detail below.

Batch Size

The batch size is considered since it greatly impacts how the network estimates the gradient of the loss function. The bigger the batch size, the better the estimate of the gradient is, which in turn makes for a more stable training. However, the disadvantages of using a large batch size are that it consumes more memory. Furthermore, a large batch size results in slower training since the network needs more samples to estimate the gradient before updating the weights. On the other hand, using too small of a batch size can result in unstable training due to the network having fewer samples to estimate the gradient.[75]

Epoch

An epoch is when all the training data have been presented to the network once. Choosing the number of epochs a network should train for is not simple. It is complicated as the network might train for too long and hence start overfitting. However, if the number of epochs is too small, the network might not train enough and cannot make robust predictions.[76] Commonly used strategies for early stop training is listed below:

- Stop training if validation loss increases
- Stop training if performance decreases or reaches a plateau
- Stop training if loss reaches a plateau

The strategy of early stop is to prevent training too much as the model would start overfitting, meaning that the network will not be able to generalize to new data. The training can be monitored by checking if the validation loss increases or if the performance accuracy decreases. Another good approach to avoid overfitting the network is to periodical

save the model during training. If the model overfits to the training data, it is possible to retrieve an older model, which might generalize better to new data.[76]

Figure 4.4 shows the training of GG-CNN on the Graspnet data-set over 80 epochs. The graphs show that the IOU performance of the network does not increase after epoch 23 as well as the training and validation loss seemingly reach a plateau at this point. This indicates that the network starts to overfit the network to the training data.

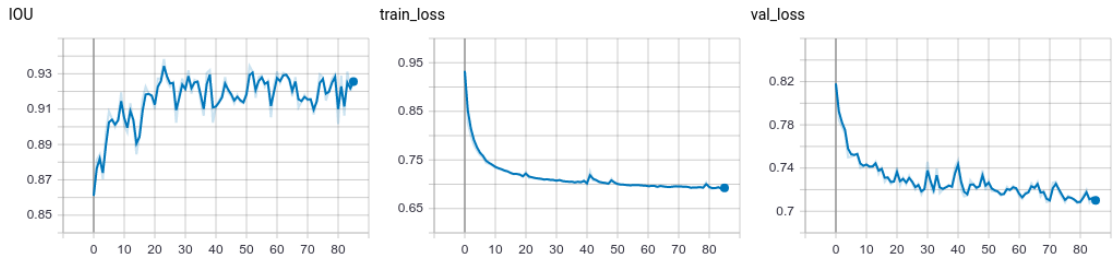


Figure 4.4: Performance and loss curves from training GG-CNN on the Graspnet data-set for 80 epochs. ***Left****: The IOU metric performance on 10% validation set.* ***Middle****: Training loss.* ***Right****: Validation loss.*

Optimizers

Another aspect to consider during training is optimizers, which are essential when training a neural network. They are used to change the weights of the network and the learning rate to accelerate learning.

Optimizers have been researched in recent years. Many optimizers exist, each having advantages and disadvantages[77]. One commonly used optimizer in grasping neural networks is Adam. It is chosen for training during this project.

Adaptive Moment Estimation (Adam) expands on the idea of Stochastic Gradient Descent (SGD) with momentum.

Stochastic Gradient Descent (SGD) solves the problem of the traditional Gradient Descent method, where the entire data-set is used to update the weights, whereas SGD updates the weights in batches. An addition to SGD is the use of momentum, which uses the average gradient direction over time. The learning rate increases when the gradients point the same direction and reduce the learning rate for gradients with changing directions. This method has the advantage that it uses batches to update the weights, which reduce memory usage. Furthermore, converging to minima is faster when using momentum.[78]

Adam uses the same idea as introduced in SGD with momentum of considering the average past gradients, but furthermore, also consider the squared past gradients. These averages are then used to adjust the learning rate doing training. This ensures that if there is a significant change in the gradients, the learning rate would be reduced due to the change. This results in slower training compared to other optimizers, as it searches more carefully for a minimum. However, momentum might also increase learning speed if the gradients keep point towards the same direction. Adam has the same advantages that SGD has, and it converges faster to a global minimum. However, the main disadvantage is that it is computationally heavy.[79]

Figure 4.5 shows the GG-CNN performance and training loss charts with Adam optimizer in orange and SGD optimizer in blue. From the figure, it can be seen that Adam is faster at reaching high performance and lower loss compared to SGD.

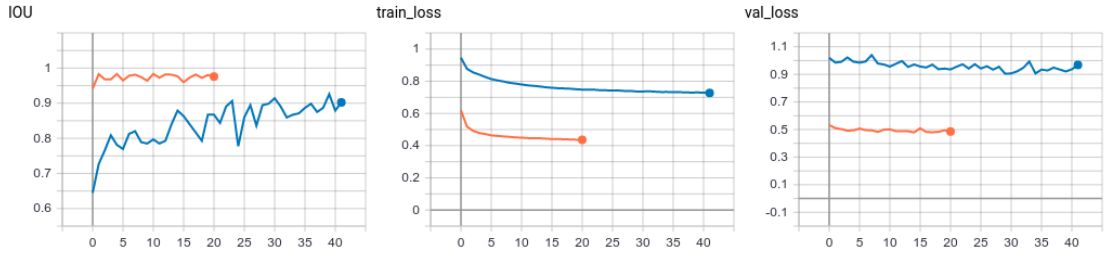


Figure 4.5: Performance and loss curves from training GG-CNN on the Graspnet data-set with Adam and SGD optimizers. ***Orange****: Adam.* ***Blue****: SGD with momentum.*

Throughout this project, the tunable parameters have been tested and evaluated through trial and error. Training the model using different parameters is evaluated in Section 5.1 in Chapter 5. For this project, the Adam optimizer has been chosen for training the networks based on the performance shown.

4.1.4 Inference

The trained network can be used for finding grasps in real-time inference. This section goes into detail on the steps needed to use the trained models for inference using GG-CNN and GR-ConvNet.

Firstly, the depth image is pre-processed by cropping the center of the image to a 300x300 pixel image. The image is then processed using inpainting as described in Section 4.2.4, where the missing values in the depth image are recovered. The processed image is then used during the forward pass of the network to predict the 3 grasp images (quality, angle, width). The three predicted images are then post-processed with a Gaussian filter, which smooths the images.

To find the best grasp from the images, the peak value at a pixel (u,v) from the quality image is found. This determines the best performing grasp location, and the coordinates can be used to index the other two images. This makes it possible to construct a grasp in the image space $g_I = (s, \theta_I, \omega_I, q)$.

Once a grasp has been found in the image space, the grasp can be converted to coordinates in the world frame for a robot to grasp the object. Knowing the extrinsic and intrinsic parameters of the camera, the grasp can be transformed into the world frame using equation 4.1, where T_C^R is a transform from the robot frame to the camera and T_I^C is a transform from the camera to the grasp in the image frame.

$$g = T_C^R T_I^C (g_I) \tag{4.1}$$

The grasp g in world coordinates is defined as $g = (p, \theta, \omega, q)$ where the position of the grasp p is given in Cartesian coordinates (X, Y, Z) in the world frame, Θ is the rotation around the Z-axis, ω is the width of the grasp in pixels. The grasp width in pixels can be converted to meters using the intrinsic parameters of the camera.

4.2 Data-set Processing and Loading

This section goes into details on how the data-sets are processed and loaded for training and evaluation. This includes cropping and resizing the images to fit the network architecture and also the details of loading the data.

The grasp generation methods chosen for the project are 2D and need input square images of size 300x300 for GG-CNN and 224x244 for GR-ConvNet. Therefore, processing of the raw data-set images is needed since they all have different sizes.

4.2.1 Cornell Data-set

To get the raw rectangular images (640x480 pixels) to a size of 300x300 or 224x224 pixels, a part of the image has to be cropped. Since each image only contains a single object, the grasping rectangles can be used for cropping the image. Furthermore, since the background of the image is noisy (the floor outside the white scene is shown), it is decided to crop out an area around the object, which will result in an image of the object on a white background.

The crop is based on the mean of the centers of the grasp rectangles. From the center, it expands 150 or 112 pixels in each direction, resulting in a 300x300 or 224x224 pixels image, as shown in Figure 4.6.

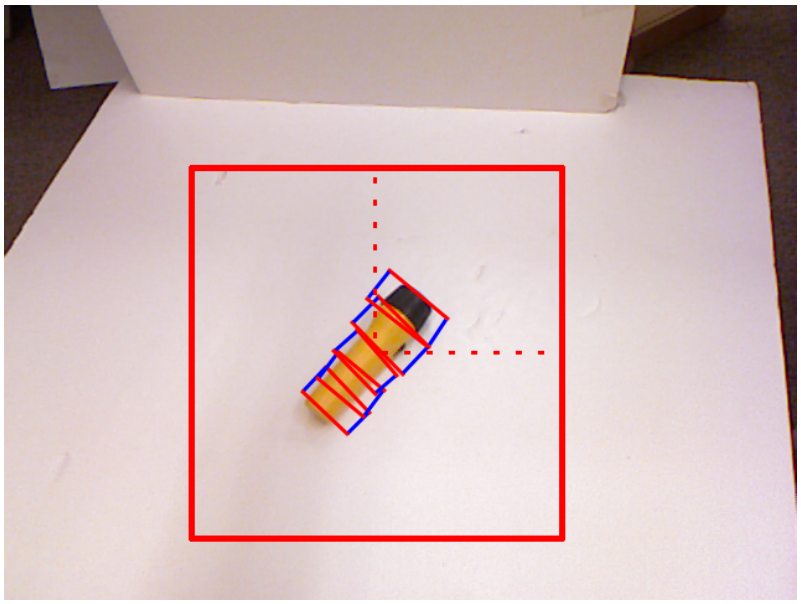


Figure 4.6: Example image from the Cornell data-set. Area outlined in red is cropped (300x300 pixels) and used for training/testing. Modified image from [47].

4.2.2 Jacquard Data-set

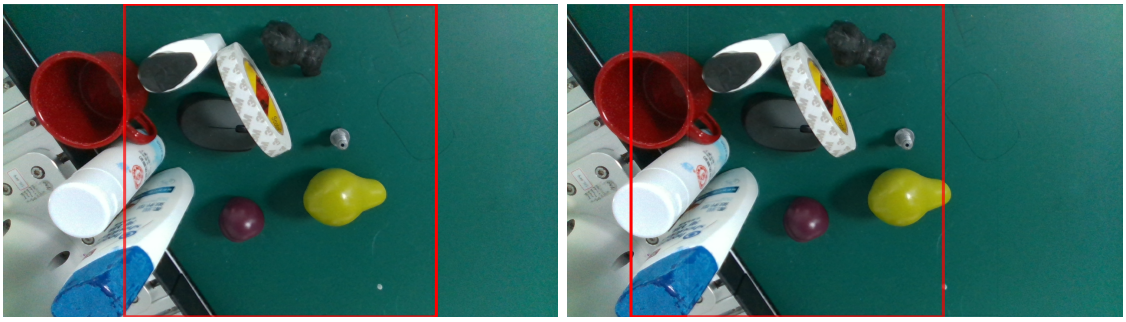
The images from this data-set are already square (1024x1024 pixels) and can be resized to 300x300 or 224x224 pixels, respectively, without the need to crop a square from the raw data. Furthermore, in contrast to Cornell, it already has a clear white background making it ideal for resizing without cropping.

4.2.3 Graspnet Data-set

To use the raw rectangular images (1280x720 pixels) from the Graspnet 1-billion data-set for training, a square needs to be cropped from the images. Since the image is rectangular and a square is needed, some data will be lost, as the objects typically are spread across the images. With around 10 objects per image, useful data will likely be lost during cropping.

Two methods can be used to crop the images. Firstly, the middle area can be cropped; this is fast but might lose unnecessary data if the objects are mostly on the side of the image, see Figure 4.7a. Secondly, a similar approach to Cornell can be used, where the crop is based around the mean center of the ground truth bounding boxes, which will keep the crop centered around most objects, as shown in Figure 4.7b.

For both methods, to maximize the preserved data, a 720x720 pixels square is cropped. The cropped area is then resized to 300x300 or 224x224 pixels.



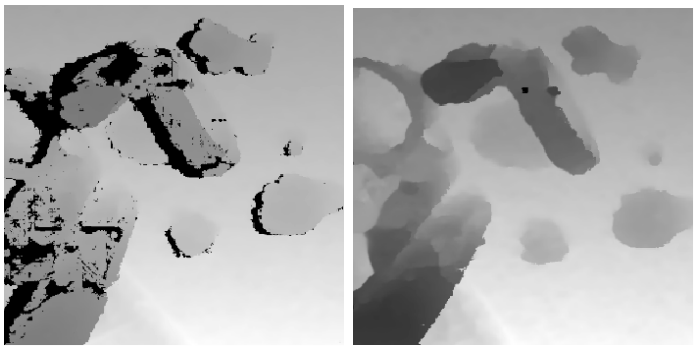
(a) Crop centered in image. *(b) Crop centered around grasps.*

Figure 4.7: Example images from the Graspnet data-set. Area outlined in red is cropped and used for training/testing. Modified images from [29].

4.2.4 Depth Image Processing

The depth images in the data-set and depth images, in general, have noisy areas due to the camera not being able to see behind the object, leaving the pixels black. Furthermore, if the camera gets too close to the object, the depth measuring sensor will not capture the depth due to hardware limitations.

When loading the depth images, they are processed. The main action taken is inpainting. A mask is created at all the pixels at or below a certain value. The pixels values are then replaced by interpolating based on the neighbouring pixels in a radius r around each masked pixel. For the images from the Cornell and Jacquard data-sets, the mask is created at pixel values of 0, which are the missing values of the depth image. For the Graspnet data-set the threshold is set at pixels with values below 0.2 m due to the hardware limitation of the camera that captured the image. An example can be seen in Figure 4.8.



(a) Before inpaint. *(b) After inpaint.*

Figure 4.8: Cropped depth image before and after inpaint. Original images from the Graspnet data-set[29].

As shown in Figure 4.8, the missing values or values that are low creates noise in the image, which is removed using the inpainting algorithm.

4.2.5 Data-set Splits

For Jacquard and Cornell, all the data is presented equally. When training, the data is split into two groups, one for training and a validation set, used to monitor the performance of the training. A split of 0.9 to 0.1 is used, where 10% of the data is set aside for testing and evaluating the model while training.

However, Graspnet has predesignated data for training and testing. Graspnet consists of scenes, which each include 256 images with annotations.

The first 100 scenes (scenes 00 - 99) of the data are used for training; these scenes are designated for training by [29].

Testing has been split into three categories:

- Objects already seen (scenes 100 - 129)
- Objects similar to training (scenes 130 - 159)
- Objects not seen before (scenes 160 - 189)

These groupings can be seen similarly to the concept of image-wise and object-wise splits seen from Cornell (more information in Section 3.3.1).

4.2.6 Ground Truth Labels

In addition to the cropping and resizing explained above, the ground truth labels are also processed to match the new image dimensions; this includes resizing the labels and offsetting them to match a crop of the original image.

One of the main differences from Cornell/Jacquard to Graspnet is the number of ground truth labels. Whereas the former has sparse labels that roughly covers the objects, Graspnet has densely labelled objects. This results in an immense amount of data, which has to be loaded when training with the Graspnet data-set. For simplicity, this difference is not tackled or handled in any way during training. However, the gap between real robot

evaluation and the rectangle metric is discussed in Section 6.2. Furthermore, the rectangle metric is discussed in Section 6.3, with one of the focus points being this difference in the number of labels.

Loading of Grasp Rectangles

When dealing with the Graspnet data-set, the load on the computer when loading the immense amounts of grasps can cause problems. This can be due to CPU bottlenecks on the system, where most of the training time is used loading the data with the CPU instead of training using the GPU. It also loads the data slow, as the amount of data itself takes a long time to read from the file and load into memory. Therefore, the amount of data can be limited, which is speculated can be done with a small impact on performance.

The number of labels can be limited by either:

- Discard based on the μ force-closure measure
- Discarding labels outside cropped image
- Simply discarding a percentage of labels
- Pre-converting and sorting of the data, so that smaller files are loaded real-time

Each grasp of the Graspnet data-set has a quality measure associated. This measure is based on force-closure. A friction coefficient μ is used to sort out grasps with bad quality. Depending on the value of μ, a different amount of labels are removed. Figure 4.9 shows the average amount of grasps per image at different values of μ.

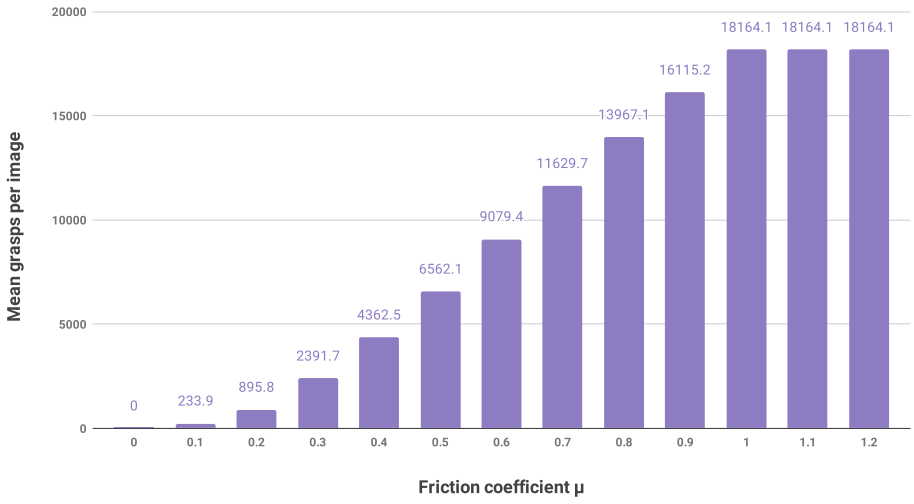
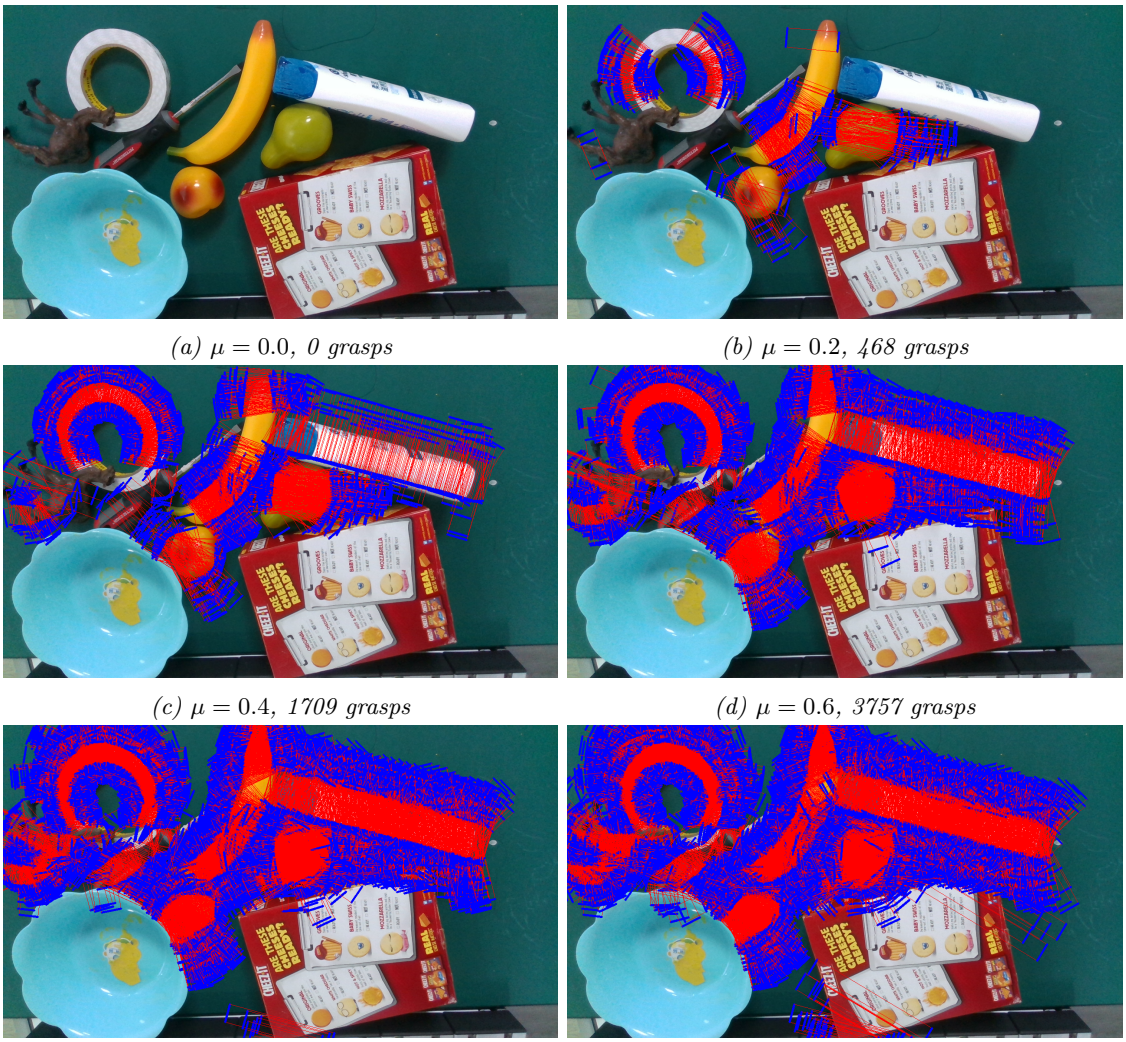


Figure 4.9: Bar chart showing the average amount of grasps per image at different values of μ.

As can be seen from the chart in Figure 4.9, choosing a higher friction coefficient μ keeps more grasps, and vice versa. In the Graspnet API[1], an example is given where $\mu = 0.2$ as a default value[2]. A visual representation from one of the images in the data-set is shown in Figure 4.10, to give an idea of which grasps are removed using the friction coefficient.

[1] Graspnet is an open-source project, for robotic object grasping
[2] `lhttps://graspnetapi.readthedocs.io/en/latest/example_loadGrasp.html`



(a) $\mu = 0.0$*, 0 grasps* *(b)* $\mu = 0.2$*, 468 grasps*

(c) $\mu = 0.4$*, 1709 grasps* *(d)* $\mu = 0.6$*, 3757 grasps*

(e) $\mu = 0.8$*, 6068 grasps* *(f)* $\mu = 1.0$*, 7645 grasps*

Figure 4.10: Scene 0, image 0 from Graspnet data-set. Grasps are sorted based on different values of μ *and drawn on the images. Modified images from [29].*

From Figure 4.10, it can be seen that the labels are not evenly distributed across the objects at different friction coefficient values. E.g. at $\mu = 0.2$, objects like the shampoo, screwdriver and toy camel have no grasps. Grasps on these objects are, however, shown at $\mu = 0.6$.

In addition to using the friction coefficient, after cropping the image, the labels outside the now cropped image are discarded. Furthermore, the grasps can be sorted by simply only considering part of the labels. E.g. only using 10% or 50% of the available labels.

The three measures can be combined in any way and are used when training different models as tuning parameters.

Finally, to alleviate some pressure on the CPU while loading the data, due to conversion between grasp representations (see Section 2.3.1), the bounding boxes are sorted and converted and saved in the new format (A format similar to Cornell). This new format can then be used in real-time when training, ultimately making training faster due to less computation in real-time.

4.3 Cloud Computing using CLAAUDIA

To deal with the vast amount of data that especially Graspnet introduces, the CLAAUDIA[3] cloud computing service at Aalborg University is used. It has powerful GPU capabilities designed for AI and deep learning.

CLAAUDIA made training faster on the Graspnet data-set since more CPU cores were available to load the data-set. CLAAUDIA allowed the use of up to 20 CPU cores and 1 GPU instead of using our home computers with a maximum of 4 CPU cores available for training. This sped up the training significantly as well as not using the entire of our own computer for training which was used for other project-related work. Only the Graspnet and Jacquard data-sets were used for training on CLAAUDIA due to the amount of data and the time required to train them. The Cornell data-set could be trained on our own computers in a few hours, whereas Graspnet and Jacquard took days.

One problem encountered and pointed out by CLAAUDIAs employee is that during training, the CLAAUDIA server used all 20 CPU cores while training but used only 1 CPU core while validating each epoch. This resulted in the validation on only 10% of the data taking longer to validate than the network training on the rest of the data-set. This problem was never resolved.

4.4 Summary

This chapter limited the selection of grasp generations methods to GG-CNN and GR-ConvNet. These methods were describe in detail, including the concept of grasp generation, the different parameters used when training and information about running inference. Furthermore, data-set processing and loading are discussed as well as details on grasp rectangles loading and processing. Finally, the cloud computing service CLAAUDIA is described. This leads to Chapter 5, where extensive evaluation and testing of the methods are performed.

[3] `https://www.claaudia.aau.dk/`

5 Evaluation

This chapter presents the main evaluations and tests conducted for the project, including training the network model using different hyperparameters. Furthermore, a custom viewpoint data-set is created to evaluate the invariance of the models and networks for different viewpoints.

5.1 Training Parameter Tuning

This section covers the results of tuning the training parameters. To find an optimal model for each architecture, the network is trained using different parameters. Each trained model can then be evaluated quantitatively using the grasp rectangle metric on different data-sets.

The model chosen for this test is GG-CNN. It is trained on the three data-sets: Cornell, Jacquard and Graspnet. Each at batch sizes: 8, 16, 32 and 64. They are all trained using Adam as the optimizer during learning. For the Cornell and the Jacquard data-sets all other parameters are default. However, for Graspnet, the image is cropped around the object's mean, as explained in Section 4.2.3. Furthermore, the friction coefficient μ is set at 0.4, and all ground truth bounding boxes are kept at this value (see explanation in Section 4.2.6).

The results from the evaluation with the rectangle metric are presented in Table 5.1. Each row shows a different set of data used for evaluation, while each column represents a model trained on a specific data-set at different batch-sizes. The Graspnet metric (see Section 3.3.2) is not used as the predicted grasps are rectangle grasps, and the Graspnet metric is based around 6-DOF grasps in point clouds.

Data-set	Cornell				Jacquard				Graspnet			
Batch size	8	16	32	64	8	16	32	64	8	16	32	64
Cornell (0.9)	86	72	82	**90**	**51**	19	11	16	5	23	**24**	9
Cornell (0.0)	-	-	-	-	**49**	27	18	24	10	23	**28**	14
Jacquard (0.9)	44	48	17	**59**	86	**88**	87	87	39	44	35	**47**
Jacquard (0.0)	44	48	18	**57**	-	-	-	-	39	45	36	**48**
Graspnet (0.9)	58	58	60	**61**	**83**	81	79	81	98	97	**99**	98
Graspnet seen	59	55	64	**66**	**78**	73	72	73	95	**96**	94	**96**
Graspnet similar	59	53	59	**64**	**78**	73	72	73	**96**	**96**	94	**96**
Graspnet novel	59	55	**65**	**65**	**78**	73	72	73	87	**90**	87	89
Mean	58	56	52	**66**	**72**	62	59	61	58	**64**	62	62

Table 5.1: Results from rectangle evaluation (%) on different data, for model trained with different data-sets at different batch-sizes. The best model per evaluation data-set split of each model from each data-set is marked in bold.

The table shows the evaluation results from the parameter tuning testing. From the average precision from each model, the best model for each data-set can be found. For training on the Cornell data-set the best accuracy is found if using a batch-size of 64. The best model training on the Jacquard data-set was trained using a batch-size of 8. Finally, when using the Graspnet data-set, the best average performance is seen using a batch-size of 16. As seen from the table, the network trained on Jacquard performers better on Graspnet than the network trained on Cornell.

5.2 Qualitative Evaluation of Q-Images During Inference

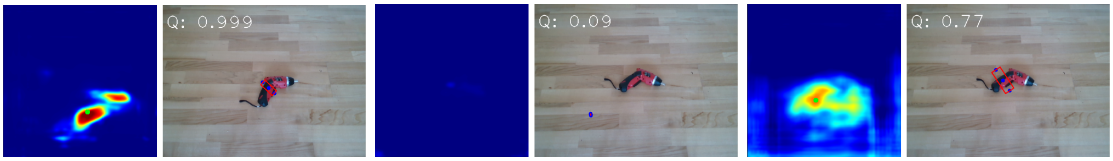
With the previous test, shedding light on the effect of the tuning parameters during training using quantitative measure, this test revolves around a qualitative measure by looking at the q-images during inference. The best performing model from Section 5.1, trained on each of the 3 data-sets, are used. This can be used to see how well the networks performs on new data in real-time and show the networks' generalisation.

Figure 5.1 shows how the network predicts grasps if no object is present in the scene. As can be seen, the 3 networks predict grasps with low grasp quality since no object is present. The network trained on Graspnet has some small artefacts around the edges of the Q image resulting in a higher grasp quality than the other two networks.

(a) Model trained on Cornell *(b) Model trained on Jacquard* *(c) Model trained on Graspnet*

Figure 5.1: Q-images and resulting grasp prediction on RGB image, for models trained on different data-sets, given an empty scene.

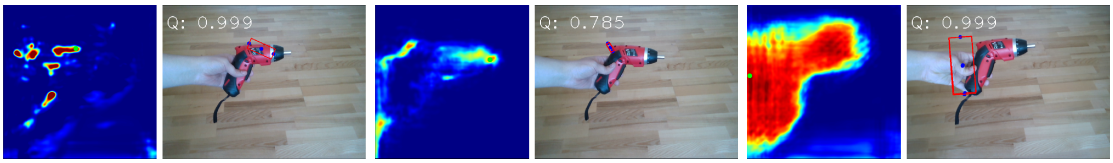
Figure 5.2 shows how well the networks predicts grasps while an object is present in the scene. The network trained on the Cornell data-set has a clean Q image while predicting a desirable grasp on the object, with limited artefacts present. The network trained on the Jacquard data-sets predicts barely any grasps while looking like no object is present in the Q image. This is speculated to be cause by the distance to the object since if the object is moved closer to the camera, the network would start predicting grasps. The network trained on Graspnet shows some artefacts in the Q image similar to that shown in the scene with no object present; however, predicts desirable grasps.



(a) Model trained on Cornell *(b) Model trained on Jacquard* *(c) Model trained on Graspnet*

Figure 5.2: Q-images and resulting grasp prediction on RGB image, for models trained on different data-sets, given a scene with a single object.

Figure 5.3 shows q-images and predicted grasps during a handover scenario. Compared to the previously presented q-images, these are noisier. Even though the q-images all show areas with high quality grasps, the resulting grasp is not ideal for any of the instances. The grasps are either in an undesired location on the object or around the arm or hand.



(a) Model trained on Cornell *(b) Model trained on Jacquard* *(c) Model trained on Graspnet*

Figure 5.3: Q-images and resulting grasp prediction on RGB image, for models trained on different data-sets, during a handover scenario.

In general, Graspnet shows promise in all the presented scenarios, despite having small artefacts. Similarly, the model trained on the Cornell data-sets shows promise, but especially in the scene with the object on the flat surface. Finally, Jacquard falls flat and is greatly outperformed by the other models.

5.3 Data-set Invariance to Changes in Viewpoint

In this test, the invariance to different viewpoints is tested. Three data-sets are created with the same scenes/objects seen from different viewpoints, namely, top, head and side, as shown earlier in Figure 2.2. It is speculated that the models trained on Cornell and Jacquard will perform better on top and head view, whereas models trained on Graspnet will be less impacted by the viewpoint due to how the data-sets are created.

When creating the viewpoints data-set, objects are picked from a pool of 15 objects, ranging from everyday objects to tools. All the objects can be seen in Figure 5.4. The data-set is freely available to download[1].

[1] `https://www.kaggle.com/runegrnhj/cornell-inspired-multiview-grasping-dataset`

Figure 5.4: The pool of 15 objects used for viewpoint test.
Including: Hammer, hand broom, soda can, Allen keys, headset, LEGO structure, saw, screwdriver, pliers, controller, remote, measuring tape, tape, pill bottle and deodorant.

The objects above are combined in scenes. The data-set consist of 50 scenes. The first 15 scenes include a single object, while the rest contains a random selection of 3 of the objects. Each object is represented in a scene a total of 8 times. Examples are shown in Figure 5.5, where a single-object scene is shown at the top and a scene with 3 objects at the bottom.

(a) Scene 1 from different views.

(b) Scene 18 from different views.

Figure 5.5: Examples of two scenes from the viewpoint data-set. Shown at the three viewpoints.

Each scene is labelled similarly to the Cornell data-set. The images can either be loaded like Cornell, where the center is cropped from the image, or like Graspnet, where the largest possible center is cropped and then resized to capture more of the scene.

The best performing models from Section 5.1 are used. That being the model trained on Cornell with a batch size of 64, the model trained on Jacquard with a batch size of 8 and the model trained on Graspnet with a batch size of 16. Each model is evaluated on the three different views to measure the relative difference in performance with scenes seen from different angles. The results are presented in Table 5.2

Data-set	**Top-view**	**Head-view**	**Side-view**
Cornell	62%	4%	10%
Jacquard	54%	20%	18%
Graspnet	62%	46%	18%

Table 5.2: Performance using the rectangle metric, for models trained on different data-set, evaluated on custom data-set at three different viewpoints.

The table shows that the top and head view has the best comparative results and that the performance on the side-view is poor. Furthermore, the model trained on the Graspnet data-set outperforms the model trained on the Cornell and Jacquard data-set.

5.4 Model Invariance to Changes in Viewpoint

To further examine the impact that the training data has, during evaluation and inference, another test revolving around the viewpoint of the training data is conducted.

Here a model is trained on one viewpoint (Top, Head or Side) of the previously mentioned custom data-set (Section 5.3). After a network is trained for each viewpoint, it is evaluated[2] on the two remaining viewpoints, to determine if it is invariant to changes in viewpoints. In addition, it is also evaluated on the viewpoint it is trained on. This would be unfair as a general evaluation metric but is shown here as a reference value to compare the other views against.

The test is conducted with two different networks, GG-CNN and GR-ConvNet.

The results are presented in Table 5.3 for GG-CNN and Table 5.4 for GR-ConvNet, where each column shows the result for a model trained on a different viewpoint. Furthermore, the diagonal showing the unfair metrics are used for comparison and are italicized.

	Trained on:		
	Top-view	**Head-view**	**Side-view**
Top-data	*78%*	58%	18%
Head-data	0%	*60%*	12%
Side-data	0%	34%	*48%*

Table 5.3: Rectangle metric results during evaluation on different viewpoints. Tested on models trained on the three different viewpoints, using GG-CNN.

[2]Using the rectangle metric

	Trained on:		
	Top-view	**Head-view**	**Side-view**
Top-data	*48%*	52%	34%
Head-data	28%	*50%*	26%
Side-data	6%	26%	*38%*

Table 5.4: Rectangle metric results during evaluation on different viewpoints. Tested on models trained on the three different viewpoints using GR-ConvNet.

From Table 5.3 it can be seen that training on data shown from the Head view is most invariant to changes in viewpoints, whereas data-sets with objects seen directly from the top lack any ability to generalize to different viewpoints. A similar pattern is shown in Table 5.4; however, the model trained on the top view images show better performance on other views using GR-ConvNet. This suggests that either the GG-CNN models are overfitted or that GR-ConvNet, in general, generalizes better to novel data. This is discussed further in Section 6.1.4.

5.5 Summary

This chapter showed the results from tests on the system. The results and the project, in general, is discussed in the following Chapter 6.

6 Discussion

This chapter discusses the report's main finding, including the results of the evaluation and other elements for discussion from, e.g. the methods chapter. Finally, it also ends by mentioning future direction for the project and possible improvements to be made that was not realized within the project's time span.

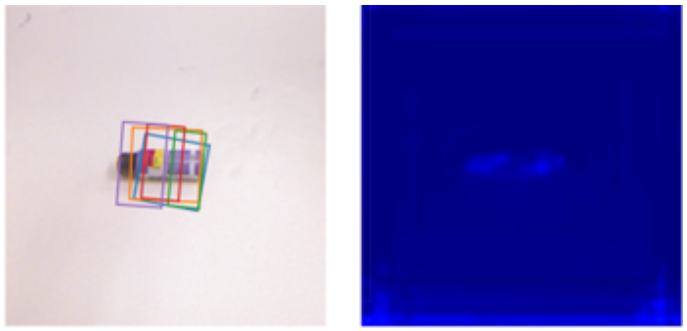
6.1 Testing/Evaluation

In this section, each of the tests in Chapter 5 are discussed. The speculations from before the tests are confirmed and denied, and the results are explained based on the findings in the tests.

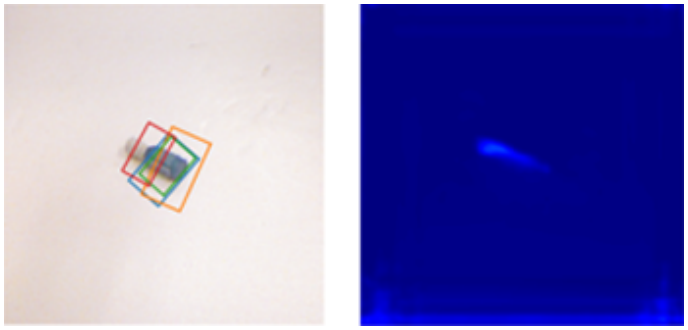
6.1.1 Parameter Tuning

Firstly, the parameter tuning test does not exhaust all possible tuning parameters. This is mainly due to the limited time of the project and helps to keep an overview of all the trained models. However, if more time was given, more parameters could be used for tuning to get a more extensive evaluation of the impact of tuning the parameters.

As shown in the test in Section 5.1, when evaluating a model trained on the Cornell data-set on the Jacquard data-set, poor performance is shown, especially the other way around. Two aspects mainly cause this. The first one is shown in Figure 6.1, where the ground truth images are shown next to the q-images. The q-image is almost entirely blue, meaning close to 0 value for the quality. This means no peak is found as a threshold is set at 0.2 to remove grasp predictions in low-quality scenes.



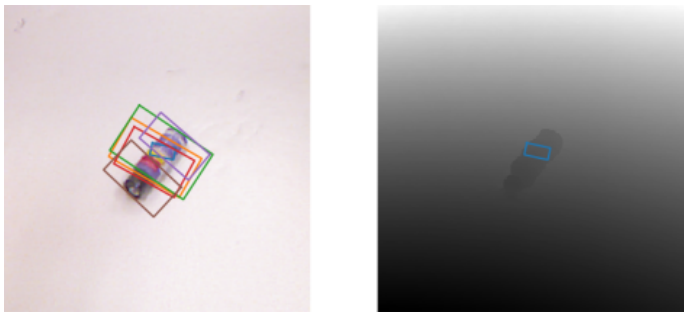
(a) First example image



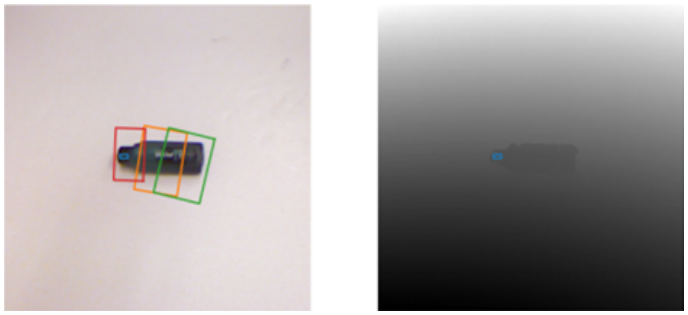
(b) Second example image

Figure 6.1: Ground truth and q image. The q-image is almost entirely blue and no grasp is predicted.

The other problem is the size of the ground truth bounding boxes having a great impact on the model's performance during evaluation. If the model is trained on small bounding boxes closely fitted to the objects, the model will be penalized during evaluation (Jaccard index metric) on data with large bounding boxes. Although the grasp might have the correct orientation and center, the model will be penalised if the gripper width is too different. Although, in a real grasping scenario, the grasp might be successful. This problem becomes apparent when training on the Cornell data-set, which has big bounding boxes, going "far" from the objects and using the Jacquard data-set for evaluation, which have smaller bounding boxes closer to the objects. Two examples are shown in Figure 6.2.



(a) Small rectangle prediction



(b) Tiny rectangle prediction

Figure 6.2: Ground truth and depth image with grasp prediction as a blue rectangle. The predicted rectangle is too small compared to the ground truth and hence it is deemed as a failed grasp.

In general, when evaluating the Graspnet data-set, similar performance is shown across all the different splits. The only exception to this is the novel objects when shown to the models trained on Graspnet. This is explained by simply being novel objects, which impacts Graspnet a bit, whereas all the splits include novel objects for models not trained on Graspnet.

6.1.2 Qualitative Evaluation of Q-Images During Inference

The images shown in the test are captured during live inference. However, only snapshots are given as a live inference video is difficult to show. The still images, however, fairly represent the general picture seen during live inference.

It can also be seen from the handover scene that the performance during handover is not ideal for implementation directly to HRI scenarios. The steps needed to be able to successfully perform human-to robot handovers are discussed in Section 6.4.

Furthermore, a test video of a handover scenario was made using the 3 chosen models to evaluate the performance in a handover scenario. The scene in the video was cluttered and noisy. The model trained on Cornell was very noisy and did not predict the grasp on the object that often. The model trained on Jacquard was better in some cases when the object was moved closer to the camera but failed when the object was too far from the camera. The model trained on Graspnet outperformed Cornell and Jacquard and predicted grasps more often on the object at multiple distances to the camera. However, as described, this test was performed in a cluttered environment, so more extensive testing should be performed, as described in Section 6.4.

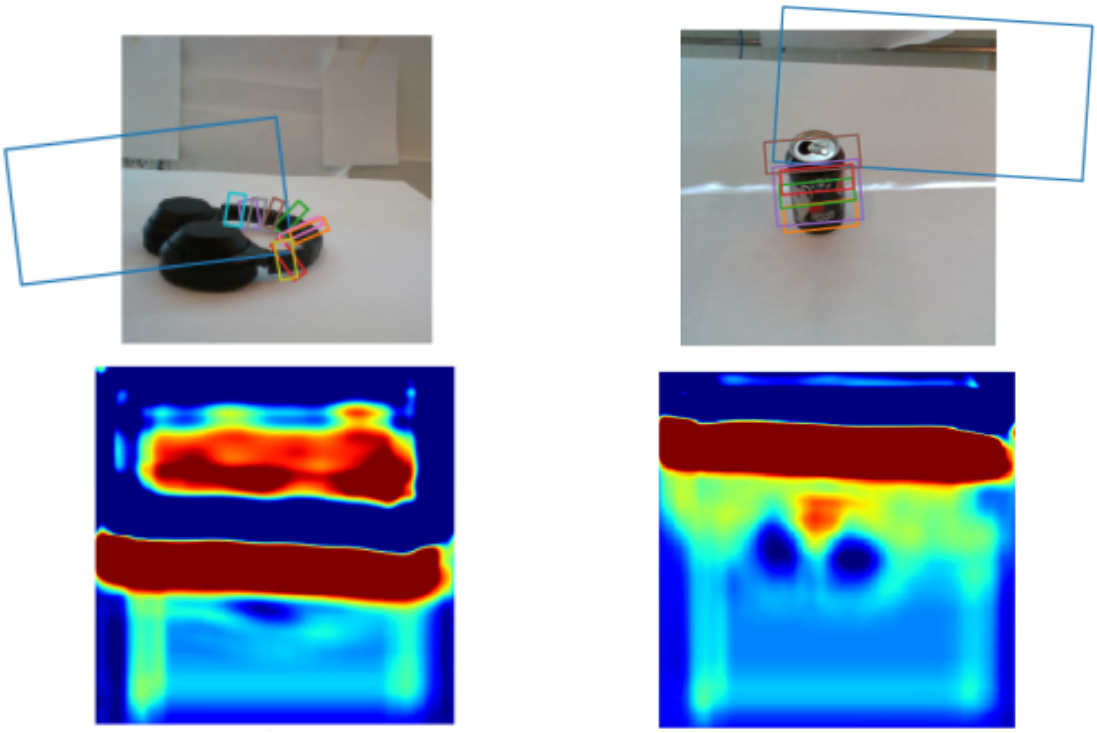
6.1.3 Data-set Invariance to Changes in Viewpoint

From Table 5.2 in Section 5.3 it is clear that models trained on the Graspnet data-set are most invariant to changes in viewpoints. This is likely due to the nature of how the data-set is captured, including many different viewpoints of the same scenes.

In general, the models have poor performance on the side viewpoint. This is to be expected at the objects are typically flat, and therefore not much of the object is captured in the image. Another problem that is found with the head and side viewpoints is the slant of the table and the depth of the background, which is explained and shown with examples in Section 6.1.4

6.1.4 Model Invariance to Change in Viewpoint

As shown in Table 5.3 in Section 5.4, when training on our custom data-set from the top viewpoint, it completely fails at generalizing to other viewpoints. Similarly, poor performance is shown in Table 5.4. This, however, can be explained using the resulting q-images during evaluation. Figure 6.3 shows some examples.



(a) Evaluation on side view *(b) Evaluation on head view*

Figure 6.3: Examples of grasping and q-images from the model trained on top viewpoint.

As can be seen in the figure, the slant of the table and the fact that the background is not all flat (as it is when viewing the object from the top) creates the illusion of an object that totally overpowers everything else on the q-image, resulting in useless grasp predictions.

This also leads to big grasping rectangles, as seen in the figure, as it attempts to grasp the seemingly huge object in the background.

6.2 Grasp Generation and Grasping

This project revolves mostly around grasp generation. However, for grasp generation to make sense in a robotic application, a physical grasp has to take place.

As grasp generation is the theoretical part of grasping, a natural gap is introduced between how the bounding box determines if a grasp is successful and how a real-life grasp would be deemed correct if it were to be performed using a robot.

In addition to the work performed and presented in the report, real robot grasping tests would be desired to conduct. It would help shed light on the gap between the grasp predictions and the real-world scenarios. Furthermore, it would indicate whether the found methods would be feasible in a handover scenario and how well it would perform compared to the theoretical results. However, this was not possible at the time of the project due to the world situation, as physical work was heavily limited.

The grasp quality measure is only an estimation of the true performance of the grasping, and therefore real grasping using a robot is needed to determine the performance of such a system conclusively. Some theoretical grasp predictions might be able to successfully grasp object using a robot.[23]

Furthermore, according to the creators of the Jacquard data-set [51], the rectangle metric can produce many "visual" false positives, that being, grasps that look bad to the human eye but is determined as good by the rectangle metric. The opposite is also present, false-negatives, where seemingly good looking grasps are predicted as bad using the rectangle metric. To deal with this problem, a simulation approach is introduced in [51], where a grasp can be evaluated in simulation in addition to the rectangle metric to determine the real performance of the grasp.

6.3 The Rectangle Metric

As presented in the previous section, Section 6.2, it is determined that there is a gap between the theoretical evaluation metric and grasping using a physical robot.

Therefore, the accuracy of the rectangle metric is questioned. It could possibly be improved to give a better representation of reality, or a new metric might be needed altogether.

One big problem encountered during the project is the difference in the number of bounding boxes between the Cornell and Jacquard data-set compared to the Graspnet data-set. The rectangle metric is designed around the Cornell/Jacquard data-sets, and is created loosely to compensate for the sparse labels[51] and is designed similarly to that of object detection (e.g. YOLO[20]). However, in grasping, the ground truth is not as well defined as in object detection. In object detection, the extremities of the object can be defined, and the bounding box can be fitted closely around the object. However, for grasping, the possibilities are almost endless, as there are copious amounts of ways to grasp one specific object.

So, when introducing a data-set with densely labelled scenes, the metric might not be strict enough for it to learn sufficiently; such a loose metric might not be needed as more of the ground truth labelled area is covered directly in the data-set by it being sparse. On another note, having a dense data-set comes with complications as more data inherently makes handling, processing and training on the data heavier, simply due to the amount of data.

On another note, when training GG-CNN or GR-ConvNet, it creates 3 ground truth images with the values of the grasp on each image. Each grasp is masked into the images one at a time (see Section 4.1.2). The problem of having densely labelled data is that many of the grasps are overlapping each other. This result in a lot of the data lost due to it being overwritten by other grasps masked onto the images, as seen in Figure 6.4.

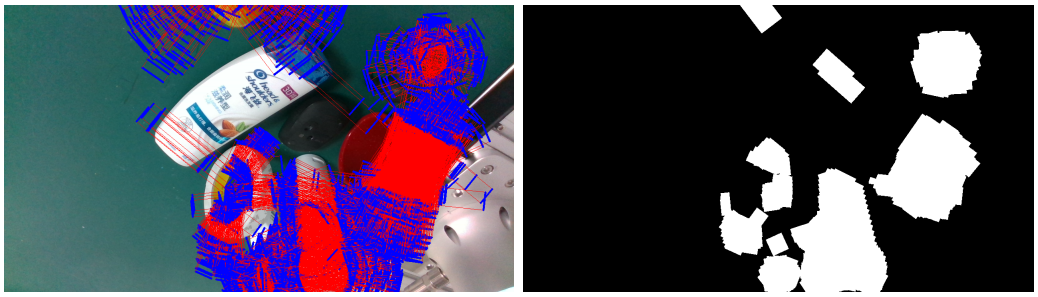


Figure 6.4: Loading of grasp rectangles. ***Left****: Raw grasping rectangles on RGB image.* ***Right****: Grasp rectangles converted to binary Q-image for training.*

6.4 Future Works

Firstly, more work could be conducted with training for the best model, changing different parameters (data augmentation, learning rate, stricter evaluation criteria, tweaking layers of models etc.) or looking into the strategy of training and opting for a different approach like, e.g. k-fold cross-validation or similar methods.

Furthermore, grasp generation has to be utilized in handover scenarios in an industrial setting.

6.4.1 Towards Handovers in an Industrial Setting

As explained in the introduction of the report, the idea of developing a robust real-time grasp generation method is to achieve human to robot handovers. Despite the promising results of the grasp generation, it is not sufficient for handover scenarios, as shown in Section 6.1.2.

Therefore, the already existing system of the LH7 robotic platform has to be utilized in combination with the grasp generation system. As explained in Chapter 1, the safety aspects are also existing in the system. Furthermore, action recognition can be utilized to initiate the handover action and start the grasp generation.

Moreover, performing grasp generation in real handover scenarios might require the removal of the hand/arm (as suggested in Section 6.1.2 and Section 5.2) to be able to achieve the final goal of human to robot handovers in an HRC/HRI setting. This is also suggested in

[14], where GG-CNN is used for human to robot handovers. However, a hand detection algorithm is used to crop out the human hand in the images to achieve grasps solely on the object in the hand.

Furthermore, in [14] a virtual plane is inserted behind the object to simulate a tabletop surface to increase the performance of the grasp generation. The reason for that is that GG-CNN is designed for grasping objects on a tabletop surface viewed from above. This virtual plane might not be needed by training GG-CNN on the Graspnet data-set since the network is trained on multiple viewpoints. Therefore, it is speculated that it might generalize better to objects in free space instead of on a tabletop surface.

Additionally, object detection or object segmentation (e.g. YOLO[19]) could be used to detect objects. The detected object could then be used as the center for cropping the image for inference. This would limit the area shown for inference and give a more focused view on the objects of importance. This concept is successfully utilized in [14].

Finally, the most important aspects are a thorough evaluation of the robotic system during handover scenarios. This includes testing and tweaking each sub-system to achieve optimal performance. In addition, a user study would be required to tune the system for HRI.

7 Conclusion

The grasp generation methods explored in this project shows potential in generating robust rectangle grasps from visual data images. Standardized conversion between grasp representations is also presented implemented in ROS and using Rviz.

The challenging problem of detecting 2D grasping rectangle in a cluttered environment is tackled. Many different methods, models, data-sets and training parameters are tried and tuned to achieve the best possible results within the span of the project. Two methods, GG-CNN and GR-ConvNet, are chosen for extensive evaluation.

As part of the evaluation, a multi-object viewpoint data-set is also introduced to shed light on the invariance to viewpoint changes of different models training on different data-set. The introduction of training on the Graspnet data-set showed better invariance to varying viewpoints than the data-sets used initially.

The grasp generation methods found at the end shows promise for use as a basis for human to robot handovers in future works as described in Section 6.4. Which mainly revolves around implementation with existing systems on the robotic platform and extensive testing and user studies.

Bibliography

[1] V. Ortenzi, A. Cosgun, T. Pardi, W. Chan, E. Croft, and D. Kuli´ckuli´c, "Object Handovers: a Review for Robotics," *arXiv*, 2020.

[2] D. W. A. Maria Bauer and M. Buss, "Human-robot collaboration: a survey," *I. J. Humanoid Robotics*, vol. 5, pp. 47–66, 2008.

[3] G. Humblot-Renaux, R. Grønhøj, and J. K. Jørgensen, "Safe robot operation in a shared workspace - human pose estimation and tracking," tech. rep., Aalborg University, Aalborg, DK, Semester Project, 2020.

[4] J. F. Buhl, R. Grønhøj, J. K. Jørgensen, G. M. Martins, D. R. Pinto, and J. K. Sørensen, "Little helper 7 - robot integration," tech. rep., Aalborg University, Aalborg, DK, Semester Project, 2018.

[5] J. F. Buhl, R. Grønhøj, J. K. Jørgensen, G. M. Martins, D. R. Pinto, and J. K. Sørensen, "Little helper 7+ - disassembly of mobile phones," tech. rep., Aalborg University, Aalborg, DK, Bachelor Project, 2019.

[6] G. Humblot-Renaux, R. Grønhøj, J. K. Jørgensen, G. M. Martins, J. K. Sørensen, and L. Væhrens, "A natural language assisted dual-arm robot system for collaborative object manipulation," tech. rep., Aalborg University, Aalborg, DK, Semester Project, 2020.

[7] D. Vogt, S. Stepputtis, B. Jung, and H. B. Amor, "One-shot learning of human–robot handovers with triadic interaction meshes," *Autonomous Robots*, vol. 42, no. 5, pp. 1053–1065, 2018.

[8] K. Strabala, M. K. Lee, A. Dragan, J. Forlizzi, S. S. Srinivasa, M. Cakmak, and V. Micelli, "Toward seamless human-robot handovers," *J. Hum.-Robot Interact.*, vol. 2, p. 112–132, Feb. 2013.

[9] A. Moon, D. M. Troniak, B. Gleeson, M. K. Pan, M. Zheng, B. A. Blumer, K. MacLean, and E. A. Croft, "Meet me where i'm gazing: How Shared Attention Gaze Affects Human-Robot Handover Timing," *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 334–341, 2014.

[10] C.-M. Huang, M. Cakmak, and B. Mutlu, "Adaptive Coordination Strategies for Human-Robot Handovers," in *2015 Robotics: Science and Systems Conference*, 2015.

[11] M. K. X. J. Pan, E. Knoop, M. Bächer, and G. Niemeyer, "Fast Handovers with a Robot Character: Small Sensorimotor Delays Improve Perceived Qualities," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[12] H. Nemlekar, D. Dutia, and Z. Li, "Prompt Human to Robot Handovers by Estimation of Object Transfer Point based on Human Partner ' s Motion," *IROS*, pp. 1–4, 2018.

[13] J. Krüger, T. Lien, and A. Verl, "Cooperation of human and machines in assembly lines," *CIRP Annals*, vol. 58, no. 2, pp. 628–646, 2009.

[14] P. Rosenberger, A. Cosgun, R. Newbury, J. Kwan, V. Ortenzi, P. Corke, and M. Grafinger, "Object-independent human-to-robot handovers using real time robotic vision," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 17–23, 2021.

[15] R. Rasch, S. Wachsmuth, and M. König, "A joint motion model for human-like robot-human handover," *2018 IEEE-RAS 18th International Conference on Humanoid Robots*, 2018.

[16] W. Yang, C. Paxton, M. Cakmak, and D. Fox, "Human grasp classification for reactive human-to-robot handovers," 2020.

[17] J. Lambrecht and S. Nimpsch, "Human Prediction for the Natural Instruction of Handovers in Human Robot Collaboration," *2019 28th IEEE International Conference on Robot and Human Interactive Communication, RO-MAN 2019*, 2019.

[18] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 5, pp. 705–724, 2015.

[19] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 1316–1322, 2015.

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[21] T. Tosun, D. Yang, B. Eisner, V. Isler, and D. Lee, "Robotic grasping through combined image-based grasp proposal and 3D reconstruction," *arXiv*, 2020.

[22] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp Pose Detection in Point Clouds," *International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[23] D. Morrison, P. Corke, and J. Leitner, "Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach," *Robotics: Science and Systems XIV (2018)*, 2018.

[24] S. Kumra, S. Joshi, and F. Sahin, "Antipodal Robotic Grasping using Generative Residual Convolutional Neural Network," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[25] S. Wang, X. Jiang, J. Zhao, X. Wang, W. Zhou, and Y. Liu, "Efficient fully convolution neural network for generating pixel wise robotic grasps with high resolution images," *IEEE International Conference on Robotics and Biomimetics, ROBIO 2019*, pp. 474–480, 2019.

[26] M. Breyer, J. J. Chung, L. Ott, R. Siegwart, and J. Nieto, "Volumetric Grasping Network: Real-time 6 DOF Grasp Detection in Clutter," *arXiv*, no. CoRL, 2021.

[27] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, 2019.

[28] J. Weisz and P. K. Allen, "Pose error robust grasping from contact wrench space metrics," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 557–562, 2012.

[29] H. S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-1billion: A large-scale benchmark for general object grasping," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 11441–11450, 2020.

[30] D. Guo, F. Sun, T. Kong, and H. Liu, "Deep vision networks for real-time robotic grasp detection," *International Journal of Advanced Robotic Systems*, vol. 14, no. 1, pp. 1–8, 2016.

[31] M. Gualtieri, A. T. Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, pp. 598–605, 2016.

[32] D. Guo, T. Kong, F. Sun, and H. Liu, "Object discovery and grasp detection with a shared convolutional neural network," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 2038–2043, 2016.

[33] Z. Wang, Z. Li, B. Wang, and H. Liu, "Robot grasp detection using multimodal deep convolutional neural networks," *Advances in Mechanical Engineering*, vol. 8, no. 9, pp. 1–12, 2016.

[34] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 3406–3413, 2016.

[35] X. Yan, J. Hsu, M. Khansari, Y. Bai, A. Pathak, A. Gupta, J. Davidson, and H. Lee, "Learning 6-DOF grasping interaction via deep geometry-aware 3D representations," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3766–3773, 2017.

[36] F.-J. Chu, R. Xu, and P. A. Vela, "Real-world Multi-object, Multi-grasp Detection," *IEEE Robotics and Automation Letters*, 2018.

[37] A. Zeng, S. Song, K. T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morona, P. Q. Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3750–3757, 2018.

[38] A. Mousavian, C. Eppner, and D. Fox, "6-DOF GraspNet: Variational grasp generation for object manipulation," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-Octob, pp. 2901–2910, 2019.

[39] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, and J. Zhang, "Pointnetgpd: Detecting grasp configurations from point sets," *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[40] H. Zhang, X. Lan, S. Bai, X. Zhou, Z. Tian, and N. Zheng, "ROI-based Robotic Grasp Detection for Object Overlapping Scenes," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 4768–4775, Institute of Electrical and Electronics Engineers Inc., nov 2019.

[41] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-DOF Grasping for Target-driven Object Manipulation in Clutter," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6232–6238, 2020.

[42] X. Lou, Y. Yang, and C. Choi, "Learning to generate 6-DoF grasp poses with reachability awareness," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1532–1538, 2020.

[43] D. Park, Y. Seo, and S. Y. Chun, "Real-Time, Highly Accurate Robotic Grasp Detection using Fully Convolutional Neural Network with Rotation Ensemble Module," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9397–9403, 2020.

[44] Y. Qin, R. Chen, H. Zhu, M. Song, J. Xu, and H. Su, "S4G: Amodal single-view single-shot SE(3) grasp detection in cluttered scenes," *Conference on robot learning*, no. 3, 2020.

[45] Y. Weng, Y. Sun, D. Jiang, B. Tao, Y. Liu, J. Yun, and D. Zhou, "Enhancement of real-time grasp detection by cascaded deep convolutional neural networks," *Concurrency and Computation: Practice and Experience 33.5*, no. June 2020, pp. 1–9, 2021.

[46] K. Chen, Y.-K. Lai, and S.-M. Hu, "3d indoor scene modeling from rgb-d data: a survey," *Computational Visual Media*, vol. 1, no. 4, pp. 267–278, 2015.

[47] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from RGBD images: Learning using a new rectangle representation," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3304–3311, 2011.

[48] T. B. Moeslund, "Geometric transformations," in *Introduction to Video and Image Processing* (I. Mackie, ed.), ch. 10, pp. 141–154, London: Springer, 2012.

[49] S. Caldera, A. Rassau, and D. Chai, "Review of deep learning methods in robotic grasp detection," *Multimodal Technologies and Interaction*, vol. 2, no. 3, 2018.

[50] D. Morrison, P. Corke, and J. Leitner, "EGAD! An evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4368–4375, 2020.

[51] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A large scale dataset for robotic grasp detection," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3511–3516, 2018.

[52] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "Shapenet: An information-rich 3d model repository," *CoRR*, vol. abs/1512.03012, 2015.

[53] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "BigBIRD: A large-scale 3D database of object instances," *Proceedings - IEEE International Conference on Robotics and Automation*, no. 5, pp. 509–516, 2014.

[54] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The columbia grasp database," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1710–1716, 2009.

[55] B. Calli, P. Abbeel, S. Member, A. M. Dollar, and S. Member, "The YCB Object and Model Set :," *International Conference on Advanced Robotics (ICAR)*, pp. 510—-517, 2015.

[56] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics," *Robotics: Science and Systems (RSS)*, 2017.

[57] J. Leitner, A. W. Tow, N. Sunderhauf, J. E. Dean, J. W. Durham, M. Cooper, M. Eich, C. Lehnert, R. Mangels, C. McCool, P. T. Kujala, L. Nicholson, T. Pham, J. Sergeant, L. Wu, F. Zhang, B. Upcroft, and P. Corke, "The ACRV picking benchmark: A robotic shelf picking benchmark to foster reproducible research," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4705–4712, 2017.

[58] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "*The Columbia Grasp Database*," 2021. `http://grasping.cs.columbia.edu/` [Accessed: 2021-4-22].

[59] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "*Bigbird Database*," 2021. `https://rll.berkeley.edu/bigbird/` [Accessed: 2021-4-22].

[60] B. Calli, P. Abbeel, S. Member, A. M. Dollar, and S. Member, "*YCB Database*," 2021. `https://www.ycbbenchmarks.com/` [Accessed: 2021-4-22].

[61] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "*shapenet Database*," 2021. `https://shapenet.org/` [Accessed: 2021-4-22].

[62] Mahler, Jeffrey and Liang, Jacky and Niyaz, Sherdil and Laskey, Michael and Doan, Richard and Liu, Xinyu and Ojea, Juan Aparicio and Goldberg, Ken, "*Dex-Net 2.0 grasping dataset*," 2021. https://berkeley.app.box.com/s/6mnb2bzi5zfa7qpwyn7uq5atb7vbztng [Accessed: 2021-4-22].

[63] J. Leitner, A. W. Tow, N. Sunderhauf, J. E. Dean, J. W. Durham, M. Cooper, M. Eich, C. Lehnert, R. Mangels, C. McCool, P. T. Kujala, L. Nicholson, T. Pham, J. Sergeant, L. Wu, F. Zhang, B. Upcroft, and P. Corke, "*APB Database*," 2021. http://juxi.net/dataset/acrv-picking-benchmark/ [Accessed: 2021-4-22].

[64] D. Morrison, P. Corke, and J. Leitner, "*EGAD Database*," 2021. https://dougsm.github.io/egad/ [Accessed: 2021-4-22].

[65] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[66] Jiang, Yun and Moseson, Stephen and Saxena, Ashutosh, "*Cornell grasping dataset*," 2021. https://www.kaggle.com/oneoneliu/cornell-grasp [Accessed: 2021-4-22].

[67] Yan, Xinchen and Hsu, Jasmine and Khansari, Mohammad and Bai, Yunfei and Pathak, Arkanath and Gupta, Abhinav and Davidson, James and Lee, Honglak, "*Vr-grasping-101 grasping dataset*," 2021. https://sites.google.com/site/deep6dofgeoawaregrasping/ [Accessed: 2021-4-22].

[68] Depierre, Amaury and Dellandréa, Emmanuel and Chen, Liming, "*Jacquard grasping dataset*," 2021. https://jacquard.liris.cnrs.fr/ [Accessed: 2021-4-22].

[69] Chu, Fu-Jen and Xu, Ruinian and Vela, Patricio A, "*multi-object grasping dataset*," 2021. https://github.com/ivalab/grasp_multiObject [Accessed: 2021-4-22].

[70] Levine, Pastor, Krizhevsky, Quillen, "*Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection*," 2021. https://sites.google.com/site/brainrobotdata/home/grasping-dataset [Accessed: 2021-4-22].

[71] Zeng, Andy and Song, Shuran and Yu, Kuan Ting and Donlon, Elliott and Hogan, Francois R. and Bauza, Maria and Ma, Daolin and Taylor, Orion and Liu, Melody and Romo, Eudald and Fazeli, Nima and Alet, Ferran and Dafle, Nikhil Chavan and Holladay, Rachel and Morona, Isabella and Nair, Prem Qu and Green, Druck and Taylor, Ian and Liu, Weber and Funkhouser, Thomas and Rodriguez, Alberto, "*MIT grasping dataset*," 2021. https://vision.princeton.edu/projects/2017/arc/#datasets [Accessed: 2021-4-22].

[72] Zhang, Hanbo and Lan, Xuguang and Bai, Site and Zhou, Xinwen and Tian, Zhiqiang and Zheng, Nanning, "*VMRD grasping dataset*," 2021. http://gr.xjtu.edu.cn/web/zeuslan/visual-manipulation-relationship-dataset [Accessed: 2021-4-22].

[73] Fang, Hao Shu and Wang, Chenxi and Gou, Minghao and Lu, Cewu, "*Graspnet-1billion*," 2021. `https://graspnet.net/` [Accessed: 2021-4-22].

[74] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[75] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[76] S. Sinha, T. Singh, V. Singh, and A. Verma, "Epoch determination for neural network by self-organized map (som)," *Computational Geosciences*, vol. 14, no. 1, pp. 199–206, 2010.

[77] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[78] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, PMLR, 2013.

[79] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.