Improved Deep Neural Networks for Generative Robotic Grasping

PREW, WILLIAM,THOMAS

Improved Deep Neural Networks for Generative Robotic Grasping

William Prew

A Thesis presented for the degree of
Doctor of Philosophy

Department of Computer Science
Durham University
United Kingdom
December 2022

Abstract

This thesis provides a thorough evaluation of current state-of-the-art robotic grasping methods and contributes to a subset of data-driven grasp estimation approaches, termed generative models. These models aim to directly generate grasp region proposals from a given image without the need for a separate analysis and ranking step, which can be computationally expensive. This approach allows for fully end-to-end training of a model and quick closed-loop operation of a robot arm.

A number of limitations are identified within these generative models, which are identified and addressed. Contributions are proposed that directly target each stage of the training pipeline that help to form accurate grasp proposals and generalise better to unseen objects. Firstly, inspired by theories of object manipulation within the mammalian visual system, the use of multi-task learning in existing generative architectures is evaluated. This aims to improve the performance of grasping algorithms when presented with impoverished colour (RGB) data by training models to perform simultaneous tasks such as object categorisation, saliency detection, and depth reconstruction. Secondly, a novel loss function is introduced which improves overall performance by rewarding the network to focus only on learning grasps at suitable positions. This reduces overall training times and results in better performance on fewer training examples. The last contribution analyses the problems with the most common metric used for evaluating and comparing offline performance between different grasping models and algorithms. To this end, a Gaussian method of representing ground-truth labelled grasps is put forward, which optimal grasp locations tested in a simulated grasping environment.

The combination of these novel additions to generative models results in improved grasp success, accuracy, and performance on common benchmark datasets compared to previous approaches. Furthermore, the efficacy of these contributions is also tested when transferred to a physical robotic arm, demonstrating the ability to effectively grasp previously unseen 3D printed objects of varying complexity and difficulty without the need for domain adaptation. Finally, the future directions are discussed for generative convolutional models within the overall field of robotic grasping.

Declaration

The work in this thesis is based on research carried out at the Department of Computer Science, Durham University, United Kingdom. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text. This work was supported by the Engineering and Physical Sciences Research Council.

Acknowledgements

I would first like to share my sincerest appreciation my supervisors for supporting me throughout this journey, Prof. Toby Breckon for giving me this opportunity and always providing me with the help I needed, Prof. Magnus Bordewich for providing guidance and correcting my mistakes, and Dr Ulrik Beierholm for his insights into neuroscience. A special thank you to my former supervisor Dr Jason Connolly who encouraged me on this path and helped me through my undergraduate towards where I am today.

My dearest gratitude towards my family and friends for aiding me and offering their help. To my parents who shared their unwavering support and acceptance on my journey, and my friends throughout my time at Durham University who believed in me and pushed me to be better and enjoy life to the fullest.

A special thanks to my colleagues who were part of Toby's research group, including Dr Neealanjan Bhomik and Dr Yona Falinie who helped answer all my questions, Dr Grégoire Payen de La Garanderie, Dr Samet Açkay, and Dr Amir Atapour Abarghouei for introducing me and guiding me through the field, and Tom Winterbottom for offering his sound advice when it was needed.

Finally, I would like to thank my partner Bryony for helping me through every stage of my work. For easing my burdens, accepting my flaws, and being there when I needed help the most. The best part of this work was sharing it with you.

Contents

List of Figures

List of Tables

Abbreviations

CGD	Cornell Grasping Dataset
CNN	Convolutional Neural Network
Dex-Net	Dexterity Network
DoF	Degrees of Freedom
EGAD	Evolved Generative Adversarial Dataset
GAN	Generative Adversarial Network
GG-CNN	Generative Grasping Convolutional Neural Network
GR-ConvNet	Generative Residual Convolutional Network
JGD	Jacquard Grasping Dataset
IoU	Intersection over Union
MAE	Mean Absolute Error
MSE	Mean Squared Error
MTL	Multi-task Learning
MTG-CNN	Multi-task Grasping Convolutional Neural Network
NLL	Negative Loss Likelihood
RGB	Colour-image consisting of Red, Green, and Blue colour channels
RGB-D	Colour image with added depth input
SOD	Salient Object Detection
YCB	Yale - Carnegie Mellon University - Berkeley

Nomenclature

$\mathcal{G}$	All the available grasps for a given object
g	A single grasp for a given object $g \in \mathcal{G}$
g_i	A single grasp in the image frame of reference
g_r	A single grasp in the robot frame of reference
$\hat{\mathcal{G}}$	Corresponding set of ground-truth labels for a given variable, e.g. $\mathcal{G}$
I	Input image
L	Total Loss function
ℓ	Loss function component
Q	Grasp quality map output
Θ	Gripper angle output
W	Gripper width output
$N(\mu, \sigma^2)$	Normal distribution (N) with mean (μ) and standard deviation (σ)
$\mathcal{N}$	Gaussian distribution

CHAPTER 1

Introduction

The study of grasping is both a challenging problem within the fields of computer vision and robotics, as it requires an understanding of intuitive physics and object properties. This is an integral factor of broader object manipulation and is necessary for future completely autonomous robotic systems to exist as they must be able to manipulate and interact with both known and unknown objects in the environment reliably and consistently. This is a wide research area as it has applications in both domestic and industrial settings [1].

Historically, creating a robotic grasping system often relied on hand-crafted analytical features that required expert human knowledge to model. However, this type of grasping system assumed that important information such as object geometry or force analytics are known [2]. For general-purpose robotics systems (such as the robotic arms displayed in Fig. 1.1), such analytic features can be difficult and time consuming to produce [3]. This has resulted in the advent of data-driven neural networks for grasping which are based on existing experience, which can be a heuristic, or generated in simulation or on a real robot [4]. Implementations differ in the ways grasp candidates are proposed and how grasp quality is estimated. All systems tackle the same challenges however, and successful strategies must ensure grasp plans are accurate to achieve grasp synthesis [4]. These criteria are generally, that a grasp must:

- enable the given task criteria;

- generalise to new and unseen objects reliably;
- and function in a range of environments with various arm designs.

This thesis aims to contribute to the growing grasping literature by proposing improvements to a subset of supervised deep learning approaches known as generative grasping models. These types of networks generate grasp candidates directly from an input image, and display their advantage that they are capable of end-to-end training and operate fast enough to suggest multiple grasps per second [5].

These improvements target each stage of the training methodology to generalise to new objects and environments. This includes the overall network architecture (Chapter 3), a targeted loss function (Chapter 4), and finally, methods of representing good training data (Chapter 5).

1.1 Motivation

Learning to interact with the immediate environment is a crucial task for applied robotics. The ability to interact with and manipulate objects is particularly challenging beyond the predictable assembly line in unstructured settings [6]. Any robot attempting to grip and interact with an object faces uncertainty in how the object will react to touch.

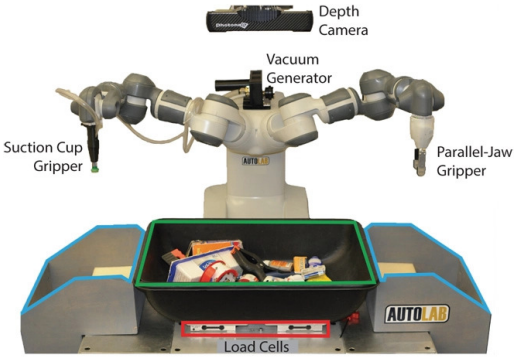
This challenge has direct application in industry with research teams at Amazon (company) hosting a yearly robotics competition to sort items from containers into categories due to the difficulty [7–9]. Attempted solutions to this task range from changes in hardware and software, with end-effectors ranging from pincer-like and multi-fingered grippers, to suction-based grippers which each attempt to pick up an object using specific grasps. Some even use hybrid grippers and choose between them depending on the best tool for the object [7, 10]. These systems take advantage of of the recent advances in machine learning to empower these robots to identify the best gripper placement, which has seen a surge of interest due to success in similar tasks such as object classification [11, 12] and semantic segmentation [13–15].

Prior surveys classify these empirical, or data-driven approaches in multiple ways, depending on the focus of the review. For example, Bohg *et al.* [4] classified these approaches based on whether they were designed to handle known, familiar, or unknown objects, whereas Kleeberger *et al.* [16] further expanded these categories as to whether they rely on model-free or model-based methods. More recently, Du *et al.* [17] split meth-

(a) WidowX [18, 19]

(b) KUKA Aiwa [20]

(c) Baxter Robot [21]



(d) Ambidextrous multi-handed robot [22]

Figure 1.1: Examples of different robot arms and grippers used.

ods according to whether they focus on object localisation, object pose estimation or grasp estimation. In practice, each of these separate classifications is true as there is a wide variety of architectures, arms, end-effectors, and sensors (see examples in Fig. 1.1). Similarly, whether these systems are trained on physical data or in simulation, which means that the systems tend to be very domain-specific and bespoke for a given task. This makes it difficult to generalise between systems and training settings as they are not invariant to changes in gripper, camera position, or field of view.

For humans, grasping unknown objects in unfamiliar environments is a trivial task by comparison, as we are able to categorise and manipulate objects quickly and effectively. Importantly, we also dynamically form grasp plans from a single egocentric view of a scene, which are problems of high interest to the robotics research community. Therefore, researchers often draw on inspiration from multiple fields such as physics, psychology, and computer science to build capable and robust robotic systems [23]. Human hands however, have the benefit of contact compliance - where the fingers mould themselves to the object and create large amounts of surface area allowing for more ambiguity when reaching [2, 24],

which is a luxury rigid robot grippers do not possess.

There has already been a large body of work which aims to employ the knowledge already understood about the way humans determine good grasp locations and apply it to these robotic systems. For example, the dual-stream hypothesis of human vision [25–27] proposes that there are two key pathways of object interaction, the ventral pathway deals with perceiving and is dedicated to object recognition [28], whereas the dorsal pathway controls gaze and eye movements for interacting with objects [29]. Machine learning research has used understood concepts from this model to attempt to improve the identification of grasps including processes such as gain-modulation [30–33], or functional grasp placement based on affordances [34–36].

Robotic grasping is still far from human levels both in terms of accuracy, speed, and dynamic reactivity [1, 23]. Our aim is to contribute towards the current state-of-the-art supervised deep learning methods, towards human levels of interaction by techniques inspired by . Currently, these supervised methods require large amounts of expertly labelled data to train initially [37], and then further amounts to transfer between domains [16], which can be expensive and time-consuming to collect even in simulation. The work presented in this thesis goes toward addressing these issues by improving the quality of the information which can be learned from existing data or reduce the overall training data required and developing a grasping system capable of success on both known and unknown objects.

1.2 Thesis Contributions

The main contributions of the thesis are as follows:

- A novel network architecture inspired by the mammalian visual system is introduced, which aims to improve the grasp success rate for unknown objects under monocular viewing conditions through multi-task learning. This Multi-Task Grasping Convolutional Neural Network (MTG-CNN) performs parallel auxiliary tasks thought to be imperative to grasping, such as: object classification; salient object detection; and a depth reconstruction task. In doing so, early layers are forced to learn representations common to both tasks. While classification negatively impacted performance, saliency and depth tasks slightly improve grasp success rates compared to base generative models on standard datasets (Chapter 3).

- A novel loss function for optimising these generative grasping models is proposed that focuses model attention by reducing the overall contribution from the background of ground truth images. The goal is to train the model to learn only the relevant grasping information at suitable locations. This generally improves grasp success rates in an offline setting compared to base generative grasping models and reduces the amount of training data required by allowing the network to converge to an optimum at a faster rate (Chapter 4).

- We also note that the widely used intersection over union (IoU) metric, often used as the state-of-the art comparison for measuring grasping success across standard datasets, is flawed. While commonly used in image datasets, it does not consider information vital for evaluating grasps such as collision data. We therefore reinforce the notion that grasping performance should be conducted with reproducible simulated or physical benchmarks alongside the often reported standalone offline data (Chapter 5).

- Finally, a Gaussian ground truth grasp representation is proposed for training the models in prior chapters to improve grasp placement by being better centred around the object. This subsequently improves grasp success in a simulated environment, and demonstrates an ability to generalise to a previously unseen physical dataset on different hardware without the need for transfer learning (Chapter 5).

1.3 Publications

The work contained within this thesis has been previously published in the following peer-review publication by the author, and is used in the chapters indicated below:

- **Improving Robotic Grasping on Monocular Images via Multi-Task Learning and Positional Loss**, W. Prew, T. P. Breckon, M. Bordewich, U. Beierholm, In Proc. International Conference on Pattern Recognition (ICPR), 2020, pp. 9843-9850 (Contributing to Chapter 3 and Chapter 4): `github.com/wtprew/mtgrasp`.

- **Evaluating Gaussian Grasp Maps for Generative Grasping Models** W. Prew, T. P. Breckon, M. Bordewich, U. Beierholm, In Proc. International Joint Conference on Neural Networks (IJCNN), 2022 (Contributing to Chapter 5): `github.com/wtprew/grasping_robot`.

1.4 Thesis Scope and Structure

The pipeline of planning a grasp involves multiple processing stages: grasp detection, grasp planning, and finally grasp execution [38]. The topics presented in this thesis focus on the grasp detection or estimation step, which directly determines the grasp pose with the highest chance of success from the sensor, while the path planning and execution stages more of concern to automation than computer vision. Chapter 2 reviews how a successful grasp is achieved by comparing early analytical which relied on manually labelling suitable locations on object models, with the range of contemporary deep learning techniques for grasp estimation, including supervised, unsupervised, and reinforcement learning algorithms. This chapter classifies and contrasts these techniques, as well as the common object and benchmark datasets available for training.

Chapter 3 explores the use of applying multi-task learning (MTL) to a supervised deep CNN architecture designed for grasp pose estimation. The efficacy of introducing additional auxiliary tasks relevant to grasping is evaluated according to a commonly used offline grasping benchmark. Additional tasks are tested against one another and hyperparameters for learning are optimised on common task representations, including changing the number of shared layers between tasks.

Chapter 4 proposes a new loss function for commonly used generative grasping models which ignores irrelevant information not required during learning. The loss function improves successful grasp rates with no drawback, without negatively impacting inference speeds or overall training time. This also helps to generalise to other settings and unseen objects, even across datasets.

Chapter 5 looks at the current evaluation metrics for model comparison. Many studies have relied on these metrics to report findings without further testing on physical arms, which can lead to inaccuracies and consequences when applied to the whole grasping pipeline. This chapter therefore proposes an updated method for training these generative models that better reflects real-world performance, by rewarding the model for centring the gripper at ground truth grasp locations. This also reinforces the use of simulated and physical grasping benchmarks, and demonstrates how the algorithms proposed in this thesis can be effectively applied and transferred to other settings and arms.

Chapter 6 evaluates the techniques presented together and provides a discussion on the overall contribution to the wider literature together with potential future directions in the field.

CHAPTER 2

Literature Review

For multi-purpose robots, grasping is an essential tool for completion of any task that requires dexterous manipulation of an object. A grasp in this case describes the process of gripping a desired object using an end-effector [39], such as fingers (in humans) or a gripper, and lifting the object successfully [3]. This is an especially difficult task in unstructured environments, changes in illumination, complex backgrounds, and inter-object occlusion. Grasp generation approaches that have improved robustness to these changes have applications to both industrial and domestic settings [1]. In each case, the system must be able to manipulate previously seen objects, as well as unknown objects, both of which pose their own fundamental challenges.

Early work with robots focused on generating individual bespoke plans for known objects, which required an expert knowledge of the robot for the specific task [2]. However, unseen objects and unstructured environments still remain a challenge, as the process is impractical particularly on large scales [40].

The overall process of the robotic grasping system can be broken down into three distinct subsystems: the grasp detection system, the grasp planning system, and the control system [38] (see Fig. 2.1 for an overview). More recently, contemporary data-driven approaches have greatly improved this first grasp detection process, and have enabled more sophisticated and accurate systems capable of generalising better to unknown objects in a variety of environments [1]. This is due to significant advancements in publicly

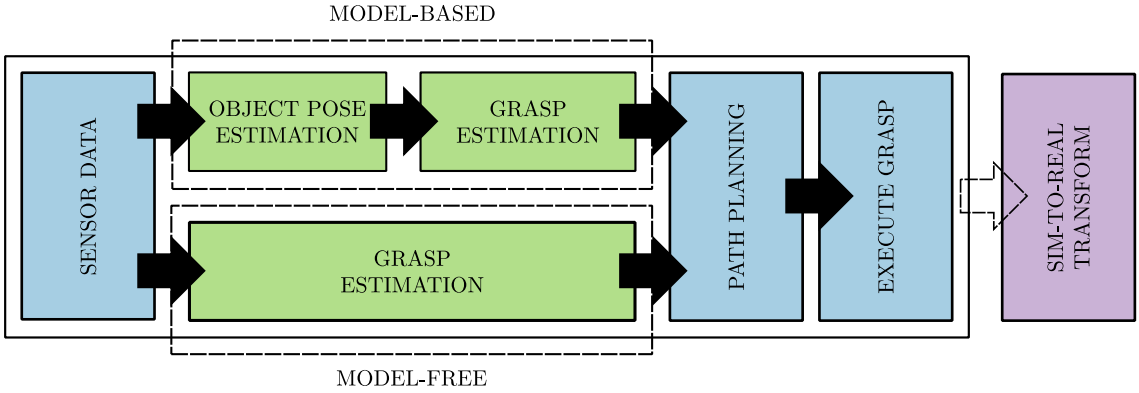


Figure 2.1: The grasping pipeline. This review generally covers the first grasping detection subsystem (green).

available data, computational resources, and improvements in machine learning grasp generation algorithms. This has also enabled deep neural networks to achieve great success in visual tasks relevant to grasping such as classification [12, 13, 41], detection [42], and segmentation [15]. A number of popular networks in these areas were in fact built upon neural networks designed for grasping including You Only Look Once (YOLO) [41], and SegNet [15].

Using these techniques, empirical models are able to infer implicit knowledge to calculate the required grasp position with the highest chance of success, including object weight, centre of mass, and material much like a humans would. Such approaches excel at reducing the extremely large set of valid grasp configurations by evaluating varying grasp contact points and gripper angles to achieve "grasp synthesis": a term which here refers to the problem of finding a grasp configuration that satisfies the relevant criteria for a given task [4]. All work covered in this review, deals with this problem either explicitly or implicitly, by either directly modelling all forces applied to the gripper, or estimating grasps with the highest chance of success by generating a large number of grasp candidates with a potential later ranking step.

To this end, a wide variety of grasping solutions exist within the literature that improve the grasping pipeline at each stage. This includes varying the implementation of machine learning algorithms such as: supervised learning [43], unsupervised learning [35, 44], or reinforcement learning algorithms [45, 46]; changing the type of hardware and gripper used including parallel plate grippers [47], multi-fingered robots [48], or suction-based grippers [5, 10]; and the type of sensors used e.g. colour (RGB) image, depth (D) image, RGB-D, point cloud, multi-view, etc. Furthermore, systems can differ in whether they operate with

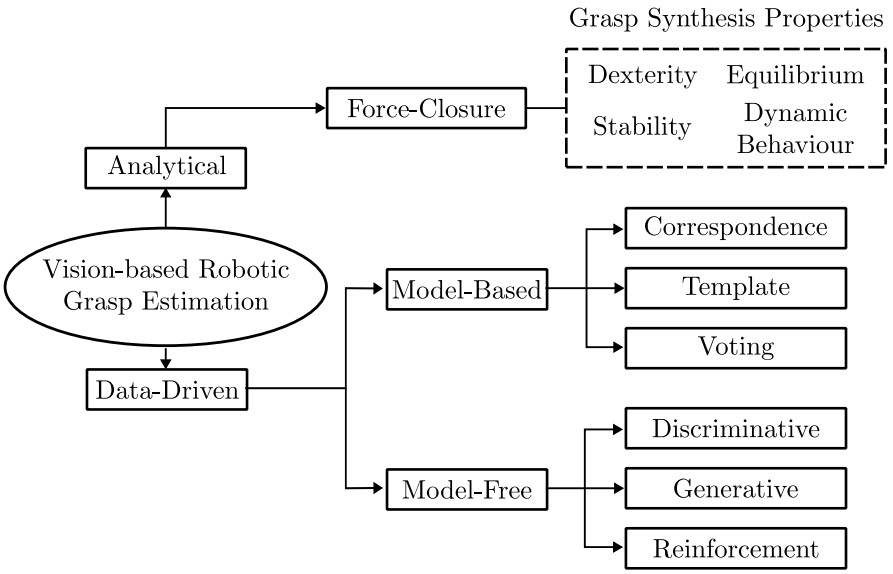


Figure 2.2: A taxonomy of different approaches to achieving grasp synthesis.

continuous feedback from visual features (closed-loop) or without (open-loop), or attempt to grasp single objects versus multiple objects on a wide variety of different benchmarks. Reacting in this way, based on continuous feedback from the input, is often referred to as visual-servoing [49]. The overall taxonomy of the techniques described in the field is described in Fig. 2.2.

The goal of this chapter is therefore, to review a wide selection of current research exploring various solutions to enable systems capable of highly accurate robotic grasping. Most of these examples tackle a similar typical robotic grasping setup as shown in Fig. 2.3 with a camera placed above the scene and a single object in view, but some also consider multiple arms or multiple objects in clutter. In aid of this, a brief overview of the early analytical (geometric) methods of robotic grasping (Section 2.1) are described and the fundamental principles required for a successful grasp to occur. This leads to an analysis of more contemporary empirical (data-driven) approaches (Section. 2.2), as well as the common publicly available datasets used to train these methods (Section. 2.3). Finally, future challenges within the field are discussed (Section. 2.4).

2.1 Analytical Approaches to Robotic Grasping

Analytic, or geometric, approaches to grasping rely on hard-coded computational algorithms for achieving grasp synthesis with an object [4,50]. This branch of research considers any approach that implements geometric, kinematic, and/or dynamic formulations in

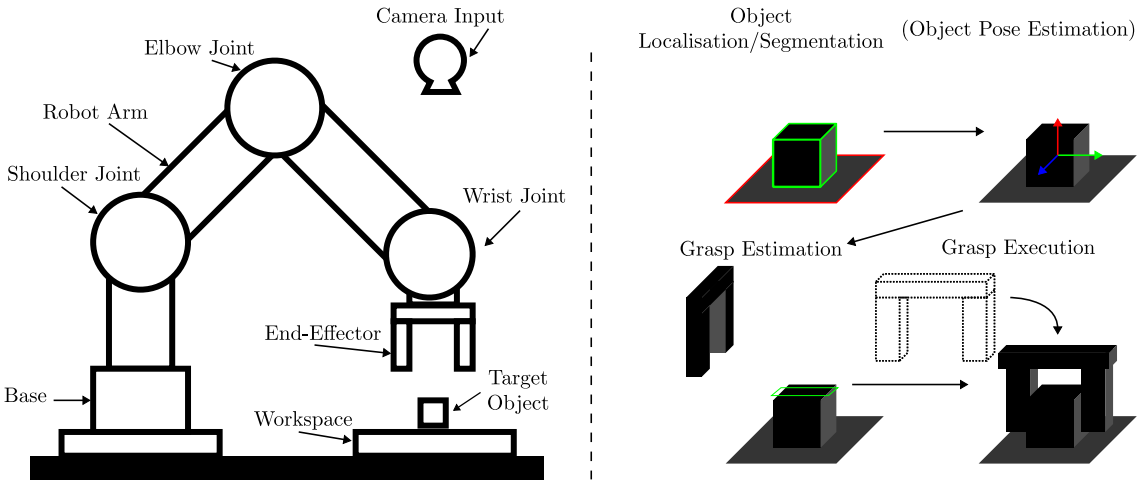


Figure 2.3: Diagram of the structure and sequence of pick and place grasping.

determining grasps. The aim of such which are to utilise multi-fingered grippers to achieve a *force-closure* grasp, a term defined in the literature as a grasp that is able to generate any external force that the grasped object may have to exert on an external body and is able to counteract any external disturbing forces that may try to loosen the grip [39].

In order to identify suitable locations for robotic fingers, research has focused on three basic approaches, mathematics, physics, and computational geometry. However, these analytical methods tend to rely on assumptions such as simplified contact models, Coulomb friction, and rigid body modelling [51,52]. Whilst these assumptions enable practical pick-and-place grasping, inconsistencies and ambiguity within the environment or noise within robot position, especially regarding grasp dynamics mean these methods are little more than approximations [4]. On the other hand, while contemporary empirical methods tend to avoid these direct computations by imitating human grasping strategies, the principles for achieving force-closure grasps and corresponding methods of grasp analysis are still relevant for any robotic grasping system.

A comprehensive analysis of early and contemporary analytical methods can be found in the review papers of Shimoga [39], Bicci & Kumar [2], and Sahbani [50], which each describe the fundamental properties needed to achieve a force-closure grasp. This can be summarised as analytical methods have aimed to generate solutions which possess the four key properties, these are: **dexterity**, **equilibrium**, **stability**, and **dynamic behaviour** [4]. Grasp synthesis is then usually formulated as an optimisation problem over the four qualities. However, there is a wide set of definitions for this terminology [53–58] and therefore the most recent definitions as described in Howard & Kumar [58] are used:

Dexterity: "This property deals with how the fingers of the robotic hand should be configured for force-closure."

Dexterous placement of these fingers is imperative for task-dependent manipulation as the planning of the fingers is also necessary for task compatibility and future planning of secondary objectives. Many configurations allow for a successful grasp, however, not all will allow for the dexterity required in a given situation.

Equilibrium: "A grasp is considered in equilibrium if: the sum of all forces and the sum of all moments acting upon the grasped object are equal to zero."

This problem deals with deciding what forces should be exerted onto the object so that the grasp is able to lift the object without damaging it [2, 39]. As a result, this property is typically seen as a force-distribution problem in determining the appropriate internal forces that should be applied. This varies on factors such as object complexity, although this can also be taken advantage of for efficient solutions, or object material [57]. In addition, this depends on factors such as the number of fingers in the end-effector, as separate algorithms were developed for n-fingered grasps [54] and multi-fingered grasps [59]. Achieving grasp equilibrium however, does not guarantee stability [60, 61].

Stability: "For a grasped object at equilibrium, the grasp is considered stable if a small disturbance on the object or end-effector does not result in dropping the object."

This is an important property when stationary, as well as during manipulation tasks. The object must return to equilibrium within the grasp once the disturbing force vanishes. Most works looking at stability in analytical works tend to focus on quasistatic assumptions and is far more difficult in dynamic settings with moving objects as forces continuously change [39].

Dynamic behaviour: "How well the robotic system reacts to changes in motion or force trajectories."

This final behaviour is only considered as a desirable property rather than an essential property, but is still important in manipulation tasks nonetheless. This is also the most difficult task as it requires fine control of each finger individually in response to large perturbations in movement and direction and therefore is difficult to remain

stable [54]. Usually, this is due to the fact that the number of forces acting upon the object is greater than the number of actuators or Degrees of Freedom (DoF) of the arm itself, meaning the arm is unable to react to every force simultaneously [2]. This is the main problem in designing control algorithms and is still a significantly difficult problem even with contemporary empirical systems.

As is the case with many analytic approaches towards grasp synthesis, many of these models are only studied in simulation where precise and accurate knowledge of hand kinematics, objects, and alignment are known [4]. In practice, noisy sensors and inaccurate models of robot kinematics, sensors, or the object can make accurate placement of the fingers difficult the same way each time. Nguyen [54] attempted to counteract this by introducing independent contact regions where the fingers can be placed independently at multiple points on an object without disturbing force-closure, and Rodriguez *et al.* [62] introduced the caging formulation which also made a manipulator with three-fingered placements more stable as they could be positioned around the object without the need for accurate positioning using waypoints. However, most of these techniques deal with rigid models of objects and assume properties such as surface properties, friction coefficients, or centre of mass and distribution of weight, which are necessary for equilibrium and stability. Therefore, most of these methods were limited to simulation [62, 63] or consider 2D objects [4, 64].

2.2 Empirical Approaches to Robotic Grasping

Empirical or data-driven approaches to robotics grasping alternatively generate grasp candidates for a given object which is ranked according to a chosen metric [4]. This is usually based on prior knowledge or experience which can be collected in simulation or using physical trials, and therefore used to be referred to as a *knowledge-based* approaches by early works [39].

Compared to analytical methods, these methods are less specific in parameterising grasps. By calculating grasps as part of an approach vector compared to specific finger-tip placements, they are therefore more robust to uncertainty in perceiving and executing grasps [4]. While this does not provide guarantees for the grasp criteria as mentioned in the previous section [39], instead estimated grasps from models are tested and verified empirically. This enables developing models which are better able to interact within

dynamic environments [4] and instead parameterise a grasp as follows according to [65,66]:

1. the grasping point on the object should be aligned with the centre of the end-effector;
2. the approach vector describing the movement the arm must take in 3-D space to reach the grasping point without obstruction;
3. the orientation or angle of the wrist joint;
4. the initial finger configuration.

Due to this simplified representation of a grasp, as well as an increase in data availability and improved hardware computing capabilities, these types of approaches have gained significant popularity in recent years. However, this remains challenging due to the wide variety of object shapes and poses they can take, and the large number of valid robot poses which can be created for solving the task. Robotic grasp implementations therefore splits the process of a grasp into a number of sub-systems [3, 38]: grasp detection (in image plane coordinates), grasp planning (mapping image to world coordinates), and arm control (inverse kinematics solution for planning).

The grasp detection sub-system is where most empirical approaches are targeted, as the grasp planning and control are usually more the focus of motion and automation disciplines [17]. With machine learning becoming a popular tool in computer vision due to comprehensive success in tasks useful to grasping, such as: classification [11], detection [41], and segmentation [15]. Deep learning techniques in particular have seen increased usage for determining successful grasp locations, although a large amount of training data is required for enabling autonomous models [37]. This data can be labelled or unlabelled depending on whether the technique uses supervised learning (SL), unsupervised learning (UL), or reinforcement learning (RL) techniques [67, 68], and labels can be collected in a simulated setting or using physical objects, either by humans or generated automatically in a self-supervised method.

The goal is to produce an automatic system, capable of being tuned to these new objects, with no expert input. Since a large amount of data is required for training such empirical models [37], it is common to generate simulated data for a greater amount of easily available training data [16], which already has access to 3D meshes, and use transfer learning to improve performance later in production (see Section. 2.3.3). Empirical approaches can be categorised into two major categories, otherwise known as model-based

or model-free, depending on the implementation and whether specific knowledge about the object is used to complete the task [16]. These methods can also be further classified based on whether they handle known, familiar, or unknown objects which Bohg [4] argued matched the dual-stream process of vision in humans. The *dorsal* stream processing action-relevant features, focused on grasp planning and object manipulation, and the *ventral* pathway related to object recognition [25, 26]. For a complete review of the neuroscience behind these processes, the reader is directed to Milner [27].

Model-based grasping is a three-stage process where the object poses are estimated, a grasp pose is determined, and a kinematically feasible, collision-free, path towards the object is planned [69, 70]. In the literature, this problem is also referred to as object pose estimation [17], which relies on prior knowledge of an object to formulate a grasp plan that estimates the transforms, such as translation and rotation, between a given reference frame and objects in the scene [69, 70]. This is particularly difficult for empirical models which have to deal with noise, occlusion, and environmental variation compared to the training data. Furthermore, these models have to deal with challenges of symmetry as different annotations can exist for the identical areas [71–73]. The main strategies employ template-matching, feature-matching (correspondence) [16], or voting-based methods [17]. This usually requires expert knowledge to manually tune an object-specific grasp configuration until grasp performance reaches a satisfactory level, however this limits generalisability towards novel or unseen objects [4].

Model-free approaches, on the other hand, directly determine grasp poses based on observations from the sensor, with the aim of better generalising to novel objects [74, 75]. This directly generates grasps from the target image by combining the object pose estimation with the grasp pose determination steps into a single grasp estimation step [16]. As the aim of this thesis is to better generalise to unseen objects based on only images without prior knowledge of the object, the rest of this review focuses on only this grasp estimation solution. For a comprehensive review of all methods of vision-based grasping, including model-based approaches, the reader is also directed towards [17].

Supervised learning approaches to model-free grasping tend to fall in one of two categories: either *discriminative* approaches sample grasp candidates which are then ranked [10, 76]; or suitable grasps are directly proposed using *generative* approaches [5, 77]. The former ranks many grasps during execution time and then chooses the grasp with the highest score. These have the advantage that many grasp poses can be evaluated but incur

a high run-time cost because they require multiple forward passes for high quality grasps to be decided. Generative models, on the other hand, generally allow for faster operating times because they only requiring one forward pass through the model with the initial highest quality grasp to be executed by the robot [7]. Reinforcement learning approaches aim to train the network with regards to an objective grasp function, letting the network learn the best locations to grasp, irrespective of the type of object without labelling from a human expert [68].

Regardless of the implementation, both methods have to contend with a number of variables to correctly identify the best way of estimating the gripper pose. The essential information for the model to calculate is the 6D gripper pose in the camera frame of reference, containing the 3D gripper position (x, y, z) and the 3D gripper orientation around those three axis (θ, ϕ, ψ). The estimation of this information either includes directly inferring the 6-DoF grasp, which requires an arm with at least as many degrees of freedom to attempt, or 2D planar grasps.

For the 2D planar grasps, the arm is constrained to a single direction and the height of the gripper (z) is fixed, grasps are then attempted on objects lying on a planar surface. This reduces the essential information to a 2D position (x, y) in the workspace, and in-plane rotation angle (Θ) to form an oriented rectangle (see Section. 2.3.1). This benefits from the improvements to deep learning applications in computer vision, which aim to produce bounding boxes of objects for detection and classification, by treating these oriented rectangles as the grasp configuration. This has allowed for the extension of grasp knowledge from known to unknown objects by evaluating the oriented rectangles [38, 42, 76–81], or directly evaluating contact points [5, 35].

6-DoF grasps extend this concept by allowing for object grasping at various angles where the 6D pose could not be simplified. This was the main focus of the analytical approaches mentioned previously (see Section. 2.1) [50], and similarly 6-DoF empirical approaches mostly aim to grasp known objects, as the placement of the gripper can be computed beforehand and then solved for the 6D pose estimation [82–84]. Deep learning have also been used as a powerful tool in this regard with some success generalising to novel objects [69, 85, 86].

To this effect, grasp estimation is conducted using a variety of inputs, with 2D planar grasps generally operating on image data which is easier to obtain. Although, this includes different types of sensor data involving RGB, D, or RGB-D image inputs, whereas 6-DoF

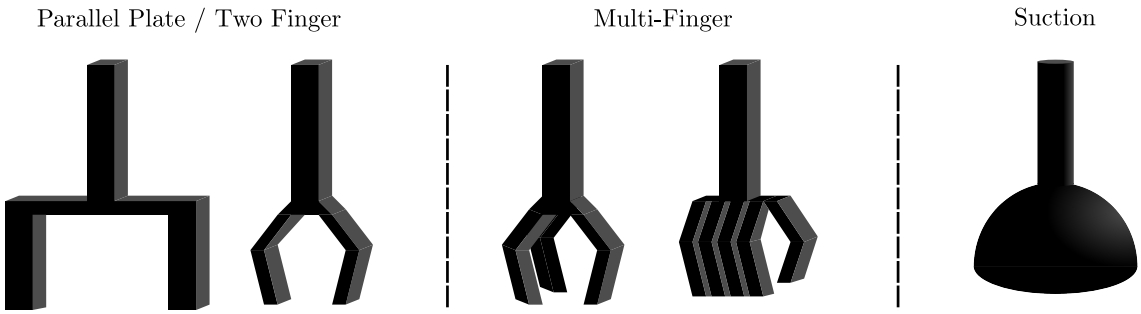


Figure 2.4: Different kinds of robotic end-effectors used.

systems generally operate on complete or partial point cloud (PC) data where the 3D shape can be reconstructed. This is further complicated by having to account for factors such as the type of gripper (end-effector) used when applying these systems, with most work using the relatively simple parallel plate gripper [76], but multi-fingered and suction-based grippers are also used [22]. Parallel grippers are generally preferred for their ease of achieving a relatively stable force-closure, however multi-fingered grippers can achieve greater dexterity [87] but generally introduce increased complexity in design. Suction-based grippers on the other hand, are generally more limited to grasping simple objects because they require a flat surface to achieve stability (example diagrams are shown in Fig. 2.4).

The rest of this section therefore compares the advantages and disadvantages of model-free grasp estimation approaches using neural networks. A comprehensive break down of the classification of methods used in the literature up until this point is featured in Table. 2.1, however, these approaches are tested using a wide range of dataset benchmarks and test objects which makes it difficult to draw a direct comparison. Some methods test on physical benchmarks, while others use transfer learning to train in simulation and generalise to real-world scenes. Therefore, the table includes reported grasp success values according to their own work, and the generalisation from training data to real-world robotic arms are then covered.

Table 2.1: Overview of model-free based approaches to robotic grasping (sorted chronologically by year of publication). Success rates are determined according generally separate and bespoke benchmarks.

	Input Data	ML Type	Training Setting	Category	Approach	End-effector	Gripper Freedom	Open-loop/ Closed-loop	Success Rate
Two Stage [78]	RGB-D	SL	Physical	Discriminative	Classification	Parallel jaw	2D	Open-loop	84-89%
SingleGrasp [77]	RGB-D	SL	Physical	Generative	Regression	Parallel jaw	2D	Open-loop	73.9%
MultiGrasp [77]	RGB-D	SL	Physical	Generative	Regression	Parallel jaw	2D	Open-loop	88.0%
Self-supervised [21]	RGB	SL	Physical	Discriminative	Classification	Parallel jaw	2D	Open-loop	79.5%
Dex-Net [76]	Depth	SL	Simulation	Discriminative	Segmentation	Parallel jaw	2D	Closed-loop	~98%
Dex-Net 2.0 [10]	Depth	SL	Simulation	Discriminative	Classification	Parallel jaw	2D	Closed-loop	80%
Google Grasp [20]	RGB	RL	Physical	Reinforcement	Reinforcement	Two finger	2D	Closed-loop	80-90%
GG-CNN [5]	Depth	SL	Physical	Generative	Regression	Parallel jaw	2D	Closed-loop	81-88%
QT-Opt [88]	RGB	RL	Physical	Reinforcement	Reinforcement	Two finger	2D	Closed-loop	76-96%
FC-GQ-CNN [89]	Depth	SL	Simulation	Generative	Segmentation	Parallel jaw	4D	Closed-loop	85-87%
QT-Opt RCAN [90]	RGB	RL	Simulation	Reinforcement	Reinforcement	Two finger	2D	Closed-loop	64-76%
Dex-Net 3.0 [22]	Depth	SL	Simulation	Discriminative	Classification	Jaw/Suction	2D	Closed-Loop	92%
GraspNet [85]	PC	UL	Simulation	Discriminative	Classification	Parallel jaw	6D	Open-loop	88%

2.2.1 Discriminative Models

Discriminative models are defined by their ability to rank grasps during execution time and then choose the grasp with the highest score, with the latter half typically performed by a neural network. This can result in carefully evaluated grasps since an arbitrary number of grasp poses can evaluated to result in the highest quality grasp [10, 22, 89]. Early examples include Lenz *et al.* [78], which was the first to apply neural networks to robotic grasping. This used a two-stage method, with the first half presenting and generating grasps using a sliding-window approach, and the second half ranking these grasps based on the colour image, the depth image, and information such as surface normals. This treated grasping as a classification problem, based on the prior success of neural networks in the field at the time, however incurred high inference times because multiple forward passes through the neural networks were required to generate the best grasps and was therefore computationally expensive.

Pinto & Gupta [79] expanded this approach by attempting to alleviate one of the large drawbacks of empirical methods up until this point, the lack of training data, by creating a dataset of 50k grasp labels, including both positive and negative examples, for model training. They initially identify regions of interest and sampling image patches, then classifying those regions according to predicted grasping angle and use a convolutional neural network to predict grasp likelihood. Similarly, Park & Chun [91] used a multi-stage classifier with spatial transformer networks which was better able to classify grasp candidates based on partial observations of objects to achieve relatively high accuracy and ten Pas *et al.* [69] used a binary classification to demonstrate the successful nature of these approaches when picking from a cluttered scene. Whilst, these classification-based methods are relatively straightforward, the sampling of a large number of grasp candidates result in rather slow inference speeds.

The Dexterity Network (Dex-Net) [76, 92] alternatively treated grasping as a segmentation problem. This discriminative approach instead uses a physics and point-cloud input to grasp objects on a workspace using a green colour, in randomised poses, to easily segment the objects using background subtraction. The outcome of the resulting grasp is then added to a dataset together with the aligned cropped depth input with an added scalar in the range of $[0, 1]$ as a measure of grasp robustness. Their Grasp-Quality Convolutional Neural Network (GQ-CNN) is trained using this dataset to predict grasp success for the given candidate to generalise to unseen objects during testing. Dex-Net 2.0 [76] further

increased the size of this dataset using analytic grasping metrics alongside the GQ-CNN to achieve satisfactory performance, and later this network was extended to ambidextrous grasping policies [10] as well as suction-based grippers [22]. However, this still took some time to fully evaluate the best grasps through this sampling and ranking-based approach, so a fully convolutional approach (FC-GQ-CNN) [89] was proposed to improve the sampling rate but also reduced the degrees of freedom from a 6-DoF system (3D grasp positon and 3D gripper orientation) to a 4-DoF system (3D position and planar orientation).

2.2.2 Generative Models

Generative grasping models differ from discriminative models in that they directly output a grasp configuration for an object in a given scene [16]. These models tend to be trained using either regression or detection-based methods, and include models which are more analogous to computer-vision tasks, where a bounding box with an associated position and width are produced which is treated as the grasp location except with an added term for orientation [42] (see Section. 2.3.1).

Here, it should be clarified that the term *generative* is used to define and differentiate the direct grasp generation method from methods that sample grasp candidates [5]. However, this is distinct from the generator models as used in generative adversarial networks (GANs) [93] as real images are used as input compared to the noise used for the GAN input. Although this type of generative model can still generate new grasp instances and provides an associated probability score at each pixel.

Redmon and Angelova [77] was one of the first to propose a generative model for grasping, simplifying the two-stage process of Lenz *et al.*, by utilising a larger regression-based convolutional neural network termed *SingleGrasp*. This method took advantage of the singular uniform method to improve accuracy, whilst also improving the speed in a grasp was proposed and then classify the object from the given RGB-D image. However, due to the large number of suitable grasp poses, another method, termed *MultiGrasp*, was also introduced which aimed to generate multiple valid grasp poses for the same image which led to the popular You Only Look Once (YOLO) architecture for object detection [41]. Kumra and Kanan [38] further improved grasp success rates by using the popular Residual network (ResNet) model [12] to perform the same regression-based task.

With larger datasets available for training, further improvements to the regression-based neural networks were developed including the Generative Grasping Convolutional

Neural Network (GG-CNN) [5] which showed how multiple grasps could be generated simultaneously from a scene. By outputting a pixel-wise representation of grasps, termed *grasp maps*, this method was able to output a grasp for every pixel of a given depth input image. These maps contained a corresponding grasp quality score, angle, and gripper width for every pixel of an image, with the pixel with the highest grasp quality score being used to reconstruct the best grasp, although a grasp could also be generated from any point on the planar object. A larger model was later introduced, which achieved even better grasp success, referred to as the GG-CNN2 [47].

Detection-based methods also exist which use the reference anchor box, an initial fixed bounding-box prediction, to assist the generation of grasp candidates [41]. With an expected size of expected grasps, regression can be further simplified [94], typically by attempting a secondary task. For example, Guo *et al.* [95] introduced a hybrid architecture with visual and tactile sensing, with an axis aligned reference box of different scales and aspect ratios. The model then classifies the orientation between discrete values alongside a quality score. Chu *et al.* further improved this model with the same idea of fixed anchor boxes, again utilising ResNet, but changed the regression model into a combination of region detection and orientation classification to again propose grasps for multiple objects. Similarly, Zhou *et al.* [80] used a end-to-end fully convolutional network with a feature extractor and multi-grasp predictor, but assigned a quality grasp score to each oriented box separately. This removed the dependency on scales and aspect ratios, predicting five regression values. Most recently, Depierre *et al.* [94] added a direct dependency between the regression and score evaluation with a novel architecture and loss function to correlate them to one another.

A common theme within many of these supervised learning approaches is instead of performing a singular grasping task, performance can be improved by solving a number of simultaneous tasks, a technique commonly referred to as multi-task learning (MTL). This is a machine learning approach that aims to help models better generalise to a given main task by using the training signals of a related task to learn an appropriate inductive bias [96]. By sharing features between multiple tasks, the network is forced to learn common representations between them that may reduce overfitting, resulting in better generalisation to the original task. MTL has been shown to improve performance on a main task when simultaneously training on a simpler auxiliary task across a wide range of disciplines [97], including computer vision (e.g. using the characteristics of the road to

predict steering direction [96] or the segmentation of images for classification in the fast R-CNN model [98]); natural language processing [99, 100]; and speech recognition [101]. For robotic grasping, system performance can be improved with the learning of simultaneous tasks which can be later combined into a single grasp proposal [43], or by solving associated tasks that contribute to successful grasp proposals, such as object segmentation [102], saliency detection [103], or semantic classification [36, 104]. This is similar to the human visual system, which separates the tasks of classification and manipulation in later cortical areas, but early stages of the visual system learn common representations of the visual scene to both tasks [25–27].

MTL can be implemented through an assortment of different functions, either separately or together in the same model; as long as the model has access to all the common representations required for both tasks. Such examples can include the cross-talk of information between different networks [100], focusing attention on a given task by learning an auxilliary training bias [96], or leveraging more specific examples like representation learning for language modelling [105]. Although, for MTL to be effective that related tasks need to share a similar optimal inductive bias [106]. As such, it is not always clear which auxiliary tasks will lead to performance increases on the main task during joint learning [97]. Some studies have attempted to bypass this problem with specifically designed multi-task losses that add an extra coefficient for training a plethora of different techniques [107].

The main advantage these generative models have over the previously discussed discriminative models, such as the GQ-CNN [76, 92], is that they are able to generate grasps at a much faster rate due to lower computational demand and only requiring a single forward pass through the neural network. Although, this may result in reduced accuracy as a result. This enables systems capable of grasping in dynamic environments using closed-loop operations, i.e. able to operate using continuous feedback mechanisms [5]. Whereas open-loop systems take the initial proposed grasp and attempt a grasp on a static object, closed-loop systems are able to react to dynamically moving objects by shifting the grasp trajectory based on real-time continuous information from the model, a technique sometimes referred to as *visual servoing* [49]. Furthermore, despite being trained on isolated images in most cases, these approaches are able to generalise to operate in cluttered environments with multiple objects with minimal decrease in performance [108].

2.2.3 Reinforcement Learning

Another common technique for data-driven robotic grasping involves model-free deep reinforcement learning. This is a specific form of supervised learning but instead of learning from a series of labelled images, the algorithm is trained on a scoring function that gives feedback informing how close the network was to the correct answer and adjusts weights based on trial-and-error functions [109][1]. Semi-supervised forms of reinforcement learning are considered to be natural ways of teaching robotic grasping agents, as toddlers often receive a few labelled training data examples from an expert figure (parent, teacher, etc.) and are then exposed to a large number of non-labelled training examples. A mixture of direct training examples preceding natural reinforcement learning in this manner can outperformed pure supervision models as the system is better able to generalise on tasks while learning from the environment [110]. Therefore, this is considered as a more ecologically valid method for robotic agents learning in the field (i.e. closer to how human agents learn), and has seen increased attention after examples like AlphaGo [111] demonstrated great success in training model agents to play games, or other models demonstrate the control of simple simulated robots [112].

The goal of reinforcement learning in the context of robotic manipulation is to detect the optimal sequence of commands for accomplishing the given task, in this case this is picking up an object [113]. The differences between RL systems therefore differ in how the task grasping task is defined and rewarded [109]. Model-free reinforcment learning (RL) grasping algorithms can be categorised into value-based methods and policy gradient methods [68]. Value-based methods construct a value function for defining a policy, which is based on Q-learning [114], while policy gradient methods are developed to find a parameterised policy to maximise a given cumulative reward based on expectation [115], such as the Deep Deterministic Policy Gradient (DDPG) [112].

These systems often go beyond pure grasp estimation, as in generative or discriminative methods, and use the current state of the robot to manoeuvre into the target pose for the best grasping position including direct use of robot kinematics from the joint sensors as a model input like position, velocity, or acceleration. For example, QT-Opt [88] is a highly influential exemplar for RL and robotic grasping rewarding a 1 for a successfully lifted object and 0 for a failed grasp. This also allows for continuous closed-loop grasping

[1]For a detailed review of reinforcement learning methods, the reader is directed towards the seminal text by Sutton & Barto [109]

as the current state of the object relative to the gripper is updated consistently, which is also demonstrated in later works [20]. This makes it an overall popular option as it demonstrates an ability to perform dynamic behaviour including object disturbances including pre-grasp manipulation strategies such as pushing and shifting, and ability to pick up novel objects. The main challenges for these systems concern sample efficiency and generalisation to novel objects or environments [68].

Sample efficiency, in the context of robotic grasping, is the problem of how much data needs to be collected in order to build an optimal policy of the given task [37]. This is more challenging than traditional supervised learning where a specific input-output pairing is used to train models because there is no clear distinction between training and testing. The time the agent spends improving the policy tends to come at the expense of the model utilisation, otherwise known as the exploration-exploitation trade-off [116]. Therefore, data collection can be an expensive process, in both time and labour costs [117]. A study by Levine *et al.* [20] demonstrated large scale end-to-end data collection and training for over 800,000 grasps on 14 robots which took over 2 months to collect. This showed that data from multiple robots could be combined for effective grasping. The model could predict grasp success based on RGB images of cluttered object scenes, and also demonstrated novel behaviours such as moving objects to improve success chance and altering the type of grip used for each object (i.e. changing between a power grip or pinch grip for objects with different materials and shapes). While the results were successful, this also demonstrates the intense requirements for physical data collection to train real-world arms, and changes in hardware setup require extensive further retraining which makes it inappropriate for general application [16, 68].

Methods to alleviate the sample efficiency problem include Hindsight Experience Replay (HER) [48] which replays gathered pick-and-place task information and augments them with different reward policies than the one initially collected. However, this is not trivially applicable to all situations due to value approximations which also makes it difficult to apply to techniques such as reward shaping. An alternative solution is *imitation learning*, which demonstrates the best place to grasp an object instead of using random initialisation for rapid learning [118, 119], and shown to be successful on large scales by "Grasping in the Wild" [120]: which used human grasping examples to collect large amounts of data quickly.

The second challenge concerns generalisation, which involves using knowledge from

the source environment to perform in a target environment, albeit to a somewhat greater degree than in supervised learning techniques as RL techniques tend to be confined to the training environment and arm used for training [121]. A popular solution is to train reinforcement learning algorithms in simulation, and use transfer learning to real-world settings, as it is easy to randomise object placement while maintaining consistent starting locations for training. An updated version of QT-Opt, RCAN [90] demonstrated how domain randomisation from simulation can lead to successful results on the same real-world robotic arm. Alternatively, another popular method involves meta-learning, colloquially referred to as learning-to-learn. Instead of training on singular tasks, the goal is to learn a variety of tasks, with the intention of training a model to learn the best method in new scenarios using a smaller number of training samples [122]. This can be combined with imitation learning to allow robots to learn new skills like reaching and pushing from grasping tasks or single demonstrations [123].

Overall, RL algorithms show particular promise in high-dimensional data problems and unstructured environments with further work. At the moment, these challenges mean that RL is not used widely in real-world manipulation tasks because of scalability, as it can be expensive to collect the data and to repair if the arm is damaged during exploration [68]. Even Levine *et al.* [20] experienced significant wear and tear on the grippers due to the significant quantities of grasping required. Another concern is safety, because it is not always predictable if the system will perform the same when transferred to new environments. This makes RL appropriate for fault tolerant tasks where high levels of failure are not as much of an issue. This is why most studies at this time still prefer to train in simulation, where it is quicker to collect large amounts of data and there is a high degree of control over environmental variables like keeping the arm starting point and camera position consistent which is more difficult in real-world settings, and consequently use some form of transfer learning to reduce the overall cost of time and effort required, as in [90]. This is a common problem across both supervised- and reinforcement-based methods and is therefore discussed in the next section.

2.3 Datasets and Benchmarks

One of the major limitations of the empirical methods discussed thus far, is that they require a large quantity of data for training [37]. This can be in the form of labelled training data by experts in the case of supervised learning (SL), or tackling the sample

efficiency and exploration-exploitation payoffs in reinforcement learning (RL) [116]. In order to assess the efficacy of these models, public datasets exist for training new models to compare between new algorithms, however, there is a large amount of variation between studies including the grasping benchmarks and the type of data used.

The aim of this section is to categorise the common methods for training supervised models of grasping and describe the methods used for evaluation. While reinforcement learning algorithms are popular in the literature, they tend to be study- and policy-specific, with each method describing their own forms of data collection and grasp success rates. Therefore, these tend to be restricted to the domain where the data was collected and will only be briefly covered in this section, except for notable examples. For a complete and comprehensive look at the range of data collection methods within the reinforcement literature, the reader is referred to the broader overview of the topic as presented by Sutton & Barto [109] or Mohammed *et al.* [124].

First, the methods for defining and representing a grasp are covered, as well as the common methods for evaluating grasp success. The common simulated and physical object datasets found throughout the literature which are used to train neural networks are then described and compared, including those with and without corresponding object meshes. Finally, since a common technique for training is to train within simulation and then generalise to the real-domain, the overall approaches to generalising between these domains are covered, including domain adaption and sim-to-real transfer methods.

2.3.1 Grasp Representation and Evaluation Metrics

Grasp detection defines the ability to recognise the points and poses required to grasp an object for a given image [10]. A grasp is generally recognised to be successful when the robotic end-effector is oriented correctly onto an object and is able to securely hold and lift the object using the gripper of choice [3], as this is the starting point for many object manipulation tasks and demonstrates some of the properties described earlier (see Section. 2.1). For the robot, as in humans, this requires knowledge of the coordinate transforms between the data from the sensor to real-world locations for grasping, as well as being able to distinguish which locations on the object would result in the highest chances for success. This requires ways of representing grasps in an image frame of reference to transform into the arm frame of reference, although there are a number of methods which exist for this purpose.

Early work represented grasps directly as contact points on images or 3D simulated mesh models, and then use analytical or empirical methods to estimate the chance at a successful grasp, through methods such as finding the optimal contact points to ensure force-closure. Dex-Net 2.0 for example [76], uses point clouds and analytic grasp metrics to plan grasps by first segmenting the scene for points of interest and using the GQ-CNN to generate multiple grasp candidates with the highest quality grasp being executed. In 6-DoF grasp poses, the grasp representation must be defined by both the position and orientation relative to the end-effector, but for 2D planar grasps, these contact points can uniquely define the gripper's grasp pose.

Early examples of 2D planar grasp representation include Saxena *et al.* [125] used a probabilistic model to infer the 3D location of a grasping point in Cartesian coordinates while considering uncertainty in the camera position. Using a multi-view approach, a singular grasp g was simplified to a small grasp point region on an image $g = (x, y, z)$, showing multiple viewing angles would help to reduce the range of possible points for grasp point inference. Zhang *et al.* [46] further simplified their point-based grasp representation to a single point on 2D plane $g = (x, y)$ using a reinforcement based approach. Evaluation of these point-based methods relies on measuring the distance between the predicted grasp centre and the ground truth grasp based on a threshold value. One drawback to these methods is that they only represent a basic approach to grasping, without any parameters for gripper orientation or how open the gripper had to be.

As a result of these limitations, a more popular form of grasp representation came about known as the *oriented grasp rectangle* representation by Jiang *et al.* [42]. Their full 3D grasp configuration was defined according to seven parameters: the 3D grasping point, 3D orientation, and opening width between the jaws of the end effector, formally defined a grasp in real-world coordinates $G = (x, y, z, \theta, \phi, \psi, l)$. However, for training a model in the image plane of reference, they reduced this to a rectangle defined by the two opposite vertices, and angle θ from the x-axis to reconstruct the grasping rectangle. This was the first study to introduce the common *rectangle metric* for evaluating and comparing predicted grasp rectangles with previously defined ground truth labels. A predicted grasp G was considered correct if the area of intersection with the ground truth grasp $\hat{G}$ was greater than 50%, i.e. $\frac{Area(G \bigcap \hat{G})}{Area(G \bigcup \hat{G}}$, but also must be within 30° degrees of orientation error.

Lenz *et al.* [78] further simplified the rectangle metric by proposing a five-dimensional representation with the assumption a good 2D grasp was able to by projected back into

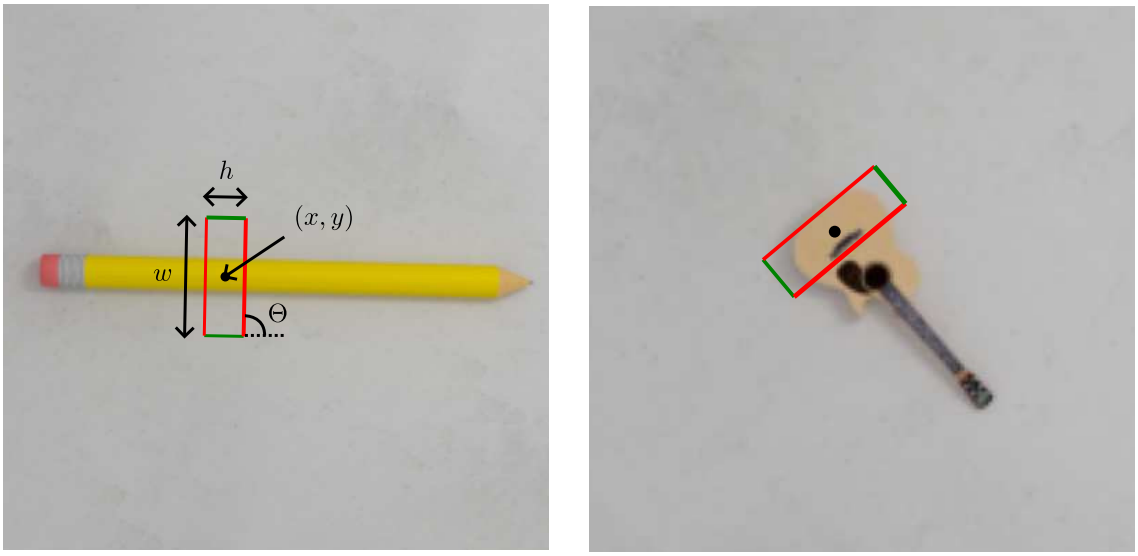


Figure 2.5: The common rectangle grasp representation [42, 78].

3D space via known transforms., which was useful for single view grasping and this made it analogous to the bounding box representation commonly used in computer vision tasks except with an added term for gripper orientation [77]. This defined a predicted grasp as $G = (x, y, \theta, h, w)$, with (x, y) representing the centre of the grasp rectangle in pixel coordinates of the 2D image, along with a gripper orientation Θ, and finally the height h and width w of the rectangle. However, they also reduced this required a minimum intersection over union (IoU) of 25% with the predicted and ground truth grasp rectangles, arguing that a single grasp rectangle can define a large space and make up most of an object. This was later empirically tested by Redmon & Angelova [77] with their SingleGrasp and MultiGrasp systems and became the main metric for evaluating neural network performance across benchmarks, e.g. [5, 38, 126]. Wang *et al.* [127] proposed a further small change which removed the height h parameter, arguing that this is a static value for a given gripper which can be controlled in the robotic set-up configurations. This gave the more commonly used four parameter grasp rectangle representation of $g = (x, y, \theta, w)$ represented in Fig. 2.5.

A final representation also seen in research, drops the gripper dimension parameters entirely in favour of only a location and orientation based grasp assumption $G = (x, y, \theta)$ [21]. The advantages of this method are that it provides a middle ground between the two previous methods, as it does not limit grasp representation to only parallel jaw grippers, whilst still proving more comprehensive data about gripper placement. Therefore, this representation is more likely to be seen in work where the gripper uses finger placements,and was

Table 2.2: List of commonly used grasp representations.

Grasp Representation		Parameters	Depth	Pose	Transform
Full Representation	[42]	$(x, y, z, \theta, \phi, \psi, l)$	Yes	Yes	No
Point Representation	[46]	(x, y)	No	No	Yes
	[125]	(x, y, z)	Yes	No	Yes
Location + Orientation	[21]	(x, y, θ)	No	No	Yes
	[128]	(x, y, z, θ)	Yes	Yes	Yes
Rectangle Representation	[78]	(x, y, θ, h, w)	No	Yes	Yes
	[127]	(x, y, θ, w)	No	Yes	Yes

further improved with a depth z coordinate for grasping in 3D space $G_z = (x, y, z, \theta)$ [128].

Of the three 2D planar grasp representations listed, the five-dimensional rectangle representation is the more common in the literature as it provides enough detail to perform training, whilst not remain over-defined, and therefore is provided along many datasets, e.g. [78, 129]. However, grasp representation is usually application specific and therefore more detailed examples are also used in some cases. For example, specific finger placements on 3-D meshes with 6D gripper poses or model-based grasp detection methods, or pixel masks where the fingers should be placed on images [130], although, these methods tend to be limited to the domain in which it was used. A summary of these representations listed can be found in Table. 2.2, with the assumption that 2D representations often require a method of transforming to 3D space. Most of these approaches are used by themselves if the aim is to pick up any given object in a scene and no requirements are necessary, but can also be used in conjunction with tasks such as object recognition to grasp identifiable objects [77], or based on other other metrics such as grasp affordances [36, 131] (i.e. grasping based on intended usage, and is akin to a segmentation problem).

2.3.2 Grasping Datasets

With the advent of data-driven grasping, new large scale datasets have been required to provide a large volume of data reserves for training these models for effective learning of task goals [74]. For supervised learning, this also requires a plethora of accurately labelled data to learn from, although this is less important in unsupervised techniques and reinforcement learning [77, 78, 109]. However, many challenges to overcome as collecting data for robotics can be expensive to acquire manually, which can be time-consuming, costly, and requiring constant supervision.

Table 2.3: List of publicly available 2D planar and 3D grasping datasets.

Dataset	Year	Reference	Type	No. Objects	No. Images	Labelled Grasps
Stanford Grasping	2008	Saxena *et al.* [74, 132]	3D simulated	10	13747	13747
Cornell Grasping Dataset (CGD)	2011	Jiang *et al.* [42, 78]	2D planar	280	1035	(8019 positive)
Carnegie Mellon University (CMU)	2015	Pinto & Gupta [21]	Reinforcement	~ 150	N/A	50567 (6266 Positive)
Yale-CMU-Berkeley (YCB)	2015	Calli *et al.* [133]	3D physical	80	Mesh	N/A
Dex-Net 1.0	2016	Mahler *et al.* [92]	3D Mesh	13k	Mesh	2.5M
Dex-Net 2.0	2017	Mahler *et al.* [76]	2D planar	~ 150	6.7M (Depth)	6.7M
Google	2018	Levine *et al.* [20]	Reinforcement	N/A	$> 10M$	800k
Jacquard	2018	Depierre *et al.* [129]	2D planar	11619	54485	1.1M

Figure 2.6: Example objects included in the YCB object dataset [133, 137].

One of the first large scale examples includes that of Pinto & Gupta [21] released the Carnegie Mellon University (CMU) dataset of 50k positive and negatively labelled grasps, although this was collected over the course of 700 hours, and the previously discussed Levine *et al.* [134] collected over 800,000 grasp attempts on 14 robots over the course of two months in order to train their reinforcement learning algorithm. However, these datasets possess a large amount of domain-specific data making it difficult to generalise to another target system or environment which can be difficult to obtain [135, 136].

Studies tend to present their results on application specific proprietary data for the best results, but this makes it difficult to compare between different algorithms and models. A major requirement for the field of robotic grasping and similar disciplines, is the need for an easily accessible benchmarks, as it is necessary to evaluate their ability to grasp both previously seen objects but also generalise to unknown objects as well as reproduce results for a required application. Therefore, it is common to use both physical and simulated benchmarks to reduce the need for data collection. Early physical benchmarks such as the Yale-CMU-Berkeley (YCB) dataset [133, 137], provide high resolution RGB-D images alongside object meshes of common household objects for training models alongside simulated trials (as shown in Fig. 2.6). However, as the aim of the dataset was to be able to be ordered, only a small sets of objects are used which make it difficult to generalise beyond them.

Another popular early dataset which contained RGB-D images of physical objects was the Cornell grasping dataset (CGD) [42, 78]. This contained an initially limited 885 images of 240 different object types, and a total of 8019 valid and invalid hand-labelled grasps. It was designed for common parallel plate grippers, and utilised the IoU metric alongside oriented grasping rectangles for evaluation and has been used extensively as a benchmark c.f. [38, 43, 95]. Although, as Mahler [76] notes each grasp was labelled by a human annotator, which introduces multiple conditionalities to the process. One of

(a) RGB Input. (b) Depth Image. (c) Positive grasps.

Figure 2.7: Example of positively labelled grasps in the Cornell Grasping Dataset [78]

which it is tedious to hand label and collect all available grasps, and therefore difficult to generalise to other objects or scenes. The other is that this process of manually labelling ground truth rectangles likely introduces human bias into the robotic grasping data as the annotator is likely to label a grasp that is easier for humans to pick up but not necessarily for a parallel plate gripper attached to a robot arm [21, 129].

With this in mind, Dex-Net 2.0 [76], instead created a synthetic dataset with 6.7M depth images with grasps labelled at the centre of each image. Whilst the previously mentioned GQ-CNN achieves high performance on this dataset, it is not possible to train models end-to-end, and therefore is limited to training discriminative models which can generate and rank grasp candidates separately. This showed that datasets could be scaled to match the training needs of empirical models and datasets are typically now labelled by the robot arms themselves, compared to the hand-labelled data from the CGD for more accurate and exhaustive ground-truth labels.

Most of these datasets have the common benefit over the CGD in that they have generated a larger amount of labelled grasps on a wider variety of objects to train data-driven models. This has been achieved through procedures such as autonomous collection from real robots on physical objects [20], or by generating a large number of grasping examples using simulated data [76, 90], in both an online fashion through reinforcement learning [18], or by using a simulated robot arm to attempt a larger number of grasps like the more recent Jacquard Grasping Dataset (JGD) [129].

The JGD also uses a simulated robot arm with various gripper widths to perform an exhaustive set of grasps on 3-D models of objects in ShapeNet [138, 139]. This features a set of 54k images and 1.1M annotated correct grasps. One major advantage of this dataset is that a simulated robot arm is provided on-line that allows performance to be tested in

(a) RGB Input. (b) Depth Image. (c) Positive grasps.

Figure 2.8: Example of positively labelled grasps in the Jacquard Grasping Dataset [129]

the same conditions as the data was generated for a more standardised benchmark, known as the Simulated Grasp Trial (SGT) score. Despite this, for speed and convenience most authors have continued to use the rectangle metric to measure performance.

2.3.3 Transfer Learning

The datasets mentioned in the previous section which are produced in simulation provide an abundant source of high-quality grasp annotations, which can be parallelised for faster training or data generation. Likewise, the quality of the gripper remains relatively high without the wear and tear of grasping objects repeatedly for high levels of control in situations where repetition is important such as reinforcement learning methods. Although, the main limitation of pre-training models using these methods becomes transferring these learned properties to the real-world on a variety of arms [16, 140, 141]. The next section therefore discusses common methods within the literature for transferring between pre-trained datasets, both physical and simulated, to new environments and arms.

For transferring between simulated to real world settings, there are three main methods for improvement: implementing better simulations which are closer to reality, or post-hoc methods such as domain randomisation [135] or domain adaptation [140, 141]. Improvements to create better simulations are generally of concern to the domain of graphics and physics engines. Examples include the early GraspIt! physics engine [142] which popularised the method of simulating grasps, but more recent and general examples used widely throughout the literature include: pyBullet [143], Blender [144], and Gazebo [145].

Domain randomisation refers to the technique of randomising elements of the simulated system, including vision observations or system dynamics, such that it trains the model to recognise the real-world as just another variation of the simulation and deal with

the variations accordingly [135]. Visual variations include changes to lighting, or object textures and colours, and system randomisations can include changing the gravity, mass of robot links, friction coefficients, or base of the robot pose [141]. This technique has been utilised by systems such as QT-Opt RCAN [90] to better improve reinforcement grasping algorithms as well as to better generalisation pick-and-place tasks [146], and segmentation tasks [147]. Another study was able to improve robotic hand manipulation by solving a Rubik's cube with multi-fingered grippers and automatically scheduling the intensity of the randomisation based on current performance [48].

Domain adaptation is a process of generalising to a target domain from a source domain, which can be achieved using unlabelled data from the target domain [16]. This applies to simulated work but also refers to transferring between arms or work environments. One method includes feature-level domain adaptation which focuses on learning domain-invariant information [140, 148], and another method focuses on pixel-level domain adaptation which restyles images similar to the target domain [90, 149]. An example of these types of domain transfer include Generative Adversarial Networks (GAN) [93], which have also previously shown to be successful in this classification of similar tasks and domains [150].GraspGAN [140] allows for a reduction in the amount of target domain data by generating real-world examples using their model. However, these approaches still require some target domain data to be successful which negatively impacts the scalability of larger models [16].

2.4 Summary

In this chapter, an overview of the current literature for autonomous robotic grasping was outlined. In Section. 2.1, the early work for identifying successful grasps and associated criteria were defined and Section. 2.2 looks at the contemporary approaches to grasping which involve training models using large quantities of data from a source domain. It was identified that for a fully autonomous robotic manipulator to be realised, in future must be able to grasp both known and unknown objects effectively, generalise from the trained source domains to target environments and arms, and operate dynamically to changes in object locations.

Data-driven approaches have been shown to achieve great progress in these areas and therefore must be able to infer grasp locations quickly and accurately. Motivated by these findings, the next chapters explore techniques to improve empirical methods and

accurately train models to recognise the relevant information for robotic grasping. Here, we focus on generative models for their fast inference speeds which enable closed-loop models of grasping, ability to generalise to new objects and environments, and their ability to be trained end-to-end. As with other neural networks, we identify three main areas that are typically targeted to improve model accuracy. Firstly, we aim to improve model performance by training networks to simultaneously perform tasks relevant to grasping such as classification, segmentation, and depth reconstruction (Chapter 3). Secondly, we introduce a novel loss function that focuses model attention to locations where grasps are more likely to succeed on improve model accuracy on unseen objects and images (Chapter 4). Finally, we assess the current metrics for evaluating offline model performance and their ability to predict performance in online grasping scenarios. We then propose an improved method for representing grasps in planar scenarios (Chapter 5).

CHAPTER 3

Multi-Task Learning for Monocular Generative Models

This chapter aims to explore whether the performance of a typical Convolutional Neural Network (CNN), trained to recognise appropriate grasps executed by a robotic arm for a given object, can be improved by introducing a related secondary task during training using a technique known as multi-task learning (MTL). Currently, these models typically underperform when trained using traditional RGB input, which are far more readily accessible when compared to the depth cameras used in some contemporary robotic grasping systems.

It has been previously shown that applying MTL has improved performance in other areas of machine learning (see Section 2.2.2), and can alleviate problems which are known to impact performance [97]. For example, it can improve training on small datasets by increasing the data availability for training when other forms of data collection may be limited [106]. MTL typically provides an inductive bias (implicit knowledge) [96] that helps generalise representations between tasks, such as by focusing attention on relevant features [106], providing implicit data augmentation [96], or reducing the risk of overfitting on data [97].

The auxiliary tasks in this chapter are chosen to provide complementary information alongside a primary grasping task, that reintroduces data intended to improve overall performance. The inspiration for these auxiliary tasks is based on the mammalian visual system, which is intended to train models to recognise inductive bias required for under-

standing a successful grasp. By understanding and sharing the general features common across both tasks, three related tasks are tested in which CNN have previously shown to achieve widespread success in, such as: object classification, salient object detection (SOD), and depth reconstruction.

The following series of experiments therefore investigate the benefits of applying MTL to a series of established generative grasping CNN. Of which, these networks have previously shown to achieve high performance at planar grasping task across established grasping datasets [5, 47]. New network architectures are then introduced that explore the effect of hard parameter sharing between distinct tasks, designed to highlight the advantage of reintroducing missing information without increasing the information required as an input. The main contributions of this chapter are therefore as follows:

- Prior generative grasping architectures are modified to introduce MTL to existing robotic grasping models, resulting in novel network architectures, before being evaluated on established grasping datasets.
- Grasping and auxiliary task performance for each network architecture is evaluated when trained with tasks expected to enhance learning, such as: object classification; salient object detection (SOD); and a depth reconstruction task.
- Additionally, this chapter provides new classification labels for the existing Cornell Grasping Dataset (CGD) [78].
- The effect of altering the number of shared parameters and corresponding weighting between each loss function between each task is investigated to highlight the impact each modification has on overall performance.

This work is then compared to base network architectures using the same methodology as the previously published studies of [5,38] on the same widely used datasets [78,129]. The effectiveness of each task is considered for appropriateness and impact on the initial robotic grasping task intended to be improved and potential future work is outlined as a result. These tasks are grouped into object classification, and SOD and depth reconstruction due to differences in methodology and intended consequence each individual task.

Most current deep learning implementations for robotic grasping tend to only focus on a single grasping task [20, 43, 77]. However, specific grasping examples have shown that there is a relative performance increase when networks are trained while performing

auxiliary tasks, such as semantic segmentation [36, 104, 151–153]; saliency detection [103]; or the generation of bounding boxes [34]. Such MTL techniques can be applied to the model using either a bottom-up [36, 103] or a top-down approach [104, 153]. Similarly, in the current study we attempt to improve overall grasping performance by forcing shared parts of the network to learn a similar auxiliary task, when presented with only monocular colour data. To this end, in the next section, a formal mathematical definition for the grasping problem we aim to solve in the subsequent chapters is given.

3.0.1 Grasping Problem

Generative grasping models are defined by their function to directly output a grasp proposal from a scene as the output of a network. Examples include early work from Redmon & Angelova [77] which introduced the single- and multi-grasp systems that popularised the grasp problem that we attempt to solve. This work defined a single planar grasp over an image as a five-dimensional output similar to Lenz *et al.* [78]:

$$g = (x, y, \Theta, h, w), \tag{3.1}$$

where grasp g is a rectangle centred around pixel (x, y), with height h and width w, and an angle of Θ. For a robot with a parallel plate end-effector (gripper), Eq. 3.1 can further be simplified as h is a constant for a given arm. In this case, the grasp pose in the robot frame of reference g_r as follows can therefore be represented as:

$$g_r = (\mathbf{P}, \Theta_r, W_r, Q), \tag{3.2}$$

with $\mathbf{P} = (x, y, z)$ being the centre of the parallel gripper jaws, Θ_r is the angle of the parallel gripper around the z-axis, W_r is the required with of the tool in mm, and Q is a value for the predicted probability that a grasp will occur, referred to as the grasp quality score.

Each generative model used here defines the grasping problem as to be able to take an n-channel input image $\mathbf{I} = \mathbb{R}^{n \times h \times w}$, which could be an RGB, D, or RGB-D image of the scene with a given object, of pixel height h and width w, and outputs a predicted 2D grasp perpendicular to a planar surface (antipodal). A single grasp rectangle in the image

frame of reference g_i, therefore consists of a position, angle and width:

$$g_i = (x, y, \Theta_i, W_i, Q), \tag{3.3}$$

where (x, y) is the centre of the proposed grasp in image pixels, Θ_i the rotation of the proposed grasp, and W_i is the required gripper width, and the subscript i indicates the values are in the image frame of reference.

However, the generative grasping networks considered in this chapter do not directly output the grasping rectangle, but output grasp for each pixel in a scene simultaneously in the form of a grasp map G. Each model outputs four per-pixel values $Q, \Theta^{\cos}, \Theta^{\sin}$, and W. Θ^{cos} and $\Theta^{\sin}$ are the two unit vectors of the gripper angles which are recombined during post-processing to form the gripper angle Θ (see Section 3.1.2) in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$ because the gripper rotation becomes symmetric around this point. This makes it easier for the network to learn compared to the angle directly according to [154]. The angle Θ may be therefore be inferred from these network outputs using Eq. 3.7:

$$\Theta = \arctan(\frac{\sin(2\Theta^{\sin})}{\cos(2\Theta^{\cos})})/2, \tag{3.4}$$

where Θ^{sin} in the range of $[0, 1]$ and $\Theta^{\cos}$ in the range of $[-1, 1]$.

Since there are four per-pixel output values, they can be viewed as images or heatmaps, and the total grasps over all pixels of an input image are represented as:

$$\mathbf{G} = (\mathbf{Q}, \mathbf{\Theta}, \mathbf{W}) \in [[0, 1] \times [-\tfrac{\pi}{2}, \tfrac{\pi}{2}] \times [0, W_{\max}]]^{h \times w}. \tag{3.5}$$

The value Q at a pixel position is the probability of a successful grasp being made centred at the location of the pixel with gripper at angle Θ and width W. Q a scalar in the range of 0 to 1, with values nearer to 1 predicting a higher chance of a successful grasp. Θ is the corresponding angle of the gripper (extracted from $\Theta^{\sin} = \sin(2\Theta)$ and $\Theta^{\cos} = \cos(2\Theta)$) required around the z-axis to grasp the object of focus and is a value in the range of $[-\frac{\pi}{2}, \frac{\pi}{2}]$ for each pixel. W_i is the width of the gripper in the range of $[0, W_{max}]$, with W_{max} being the maximum width of the parallel gripper.

To extract a single proposed grasping rectangle G_i from grasp map G, the centre of the rectangle is the pixel position equal to the maximum grasp quality Q score and use the corresponding angle Θ and width W from the same pixel position.

To execute a grasp proposal in the real world from a grasp rectangle given in image coordinates, the grasp must undergo a series of known transforms:

$$g_r = t_{RC}(t_{CI}(g_i)). \tag{3.6}$$

where t_{CI} is the transform from the 2D image coordinates into the 3D camera frame using known camera intrinsics, and t_{RC} is the transform from the camera frame to the world or robot frame (see Appendix. B for examples of these transforms).

3.1 Object Classification

The first auxiliary task considered for MTL alongside robotic grasping was inspired by the dual-processing theory of the mammalian visual system [27]. There has already been extensive research into the way humans process visual information and a number of functions can be performed independently during vision-based grasping processes and object manipulation, most notably the function of object recognition [26]. Humans can apply the knowledge from previously learned tasks to more complex systems, and some overlap between these systems must exist in order to grasp objects based on their intended function [155], i.e. must require knowledge of function and class to position end-effector/gripper/hand for a given use case.

Neural networks have already seen widespread success at classification datasets, such as on ImageNet [156], with the introduction of deep neural networks like Fast R-CNN [11] and YOLO [77], or the Common Objects in Context (COCO) [157] dataset with networks like Mask R-CNN [12]. This section therefore combines the tasks of object manipulation and classification into one network because object classification may share inherent features that enhance grasping.

3.1.1 Grasping and Classification Dataset

To address the challenge of developing a large set of training examples containing both grasp and classification labels, the choice was made to alter an existing grasping dataset with existing performance benchmarks rather than generate an entirely new dataset. As grasp labelling can be expensive in terms of human labour and financial cost, it was deemed unreasonable to create a dataset from scratch. For example, Levine et al. [20], took 3 months to collect 800,000 grasp examples leveraging up to 14 state-of-the-art robotic

arms collecting data constantly to generate sufficient training examples, which requires supervision and energy to achieve. Existing large grasping datasets are now typically formed of a large set of simulated object meshes to provide adequate training examples for grasping, such as DexNet [92], but typically do not contain corresponding classification data for objects. Those that do, generally contain only a small number of categories or objects. The Yale-CMU-Berkeley (YCB) dataset [133, 137] for example, is divided into five distinct categories, (food items, kitchen items, tool items, shape items, task items), but only exists as a physical dataset and is therefore limited to only 77 objects presenting only a small sample of grasp training data.

To reduce the need for an entirely new set of grasp labels, the existing CGD [78] was modified to add supplementary classification data for both model training and testing as this contained a sufficient amount of grasp labels and variety of objects. This dataset, initially consisting of 1035 RGB-D images of 280 real-world objects, but later reduced to only 885 images and 240 unique objects, has a total of 8019 positively labelled grasps including over 5,000 human-labelled positive grasps and nearly 3,000 invalid grasp rectangles to learn from. To provide supplementary data, artificial transformations were also used such as random cropping and rotation to increase the amount of training data and provide an increased number of grasping examples.

As the original data was collected mainly out of convenience from an office environment, it was therefore not collected with classification in mind. The dataset unfortunately features an imbalanced number of examples of the same object, with some appearing more frequently, such as stationary or objects typically present in a kitchen environment, and some objects only appear once, such as some food items.

Multiple types of class labels were initially considered that were thought to relate to grasp quality in some way. This included general properties like overall object shape, material, or texture but this lead to heavily imbalanced datasets of a singular individual type. Therefore, the dataset was labelled that aimed to encapsulate the range of objects, whilst providing enough descriptive information to help improve grasping performance.

The data was labelled according to one of two ways: a *specific* set of labels and a *general* set of labels. Specific labels included the unique name for each type of object label, such as apple, mug, or screwdriver, leading to a total of 79 unique classes of objects and the responsibility was left up to the network to infer which types of grasps worked for similar groups of objects. On the other hand, general labels categorised each object

Figure 3.1: Examples of classification labels for the training set in the Cornell Grasping Dataset [78] and the corresponding positive ground-truth grasp rectangles.

according to a given function using broader terms with similar grasp functions, like the YCB dataset [133, 137], with the intention of forcing the network to associate grasps with certain groups of objects. For example, all instances of food in the dataset, such as apples and bananas, were grouped together and the same process for other categories such as kitchen utensils or tools. This reduced the number of categories to 9 which means there more examples of each category to learn from but may make identifying categories more difficult due to the variety between objects. The full list of "specific" and "general" labels can be found in Appendix A.1 and Appendix A.2 respectively.

A number of examples objects with the corresponding valid ground truth grasp rectangles and hand-labelled categories can be seen in Fig. 3.1. Object labels are chosen as all the grasps in this dataset were labelled by human experts and was designed to train supervised learning systems. This is likely to introduce biases towards grasps that are preferred by human hands rather than parallel plate grippers [129]. For example, in Fig. 3.1, valid grasps are labelled only along the handles of objects such as in the case of the spatula, which is ideal for human tool use, whereas a parallel gripper may prefer the wider surface area of the tool part which remains unlabelled. Therefore, it is expected that valid grasp rectangles inform classification during training.

3.1.2 Methodology

Here, the methodology for the first set of experiments based on classification using the modified CGD [78] is outlined.

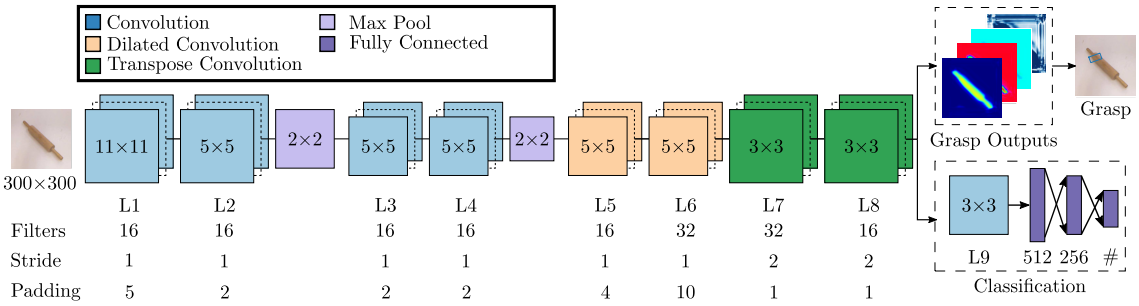
Multi-task Network Architecture

To investigate whether the effect of training a neural network to perform an auxiliary classification task can improve performance on a grasping task, our proposed solution is to introduce the multi-task grasping convolutional neural network (MTG-CNN). Based on the successful generative grasping convolutional neural network (GG-CNN) from [5, 158], the GG-CNN is a small and lightweight network for closed-loop continuous grasping that outputs a map of potential antipodal grasps perpendicular to the $x-y$ plane (as defined in Section 2.2.2). Our experiments use the follow up GG-CNN2 model architecture from Morrison *et al.* [158] as the backbone for the multi-task architecture as defined below.

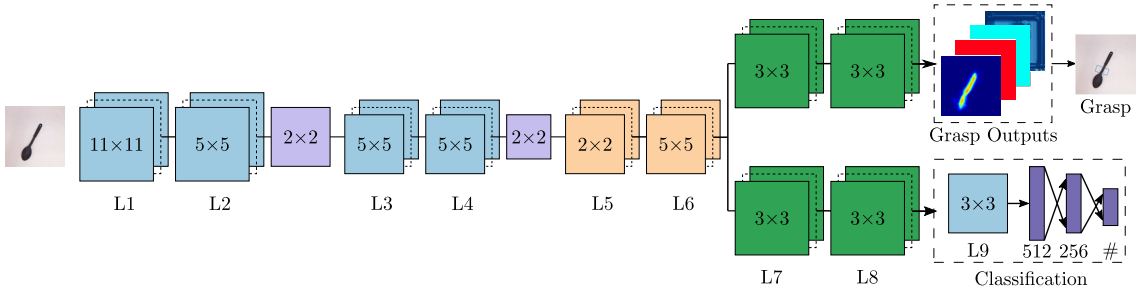
The MTG-CNN network was inspired by the dual-streams hypothesis of human vision [25–27] and extends the GG-CNN2 network by adding an extra output for the auxiliary task, in this case, for the classification of objects. To investigate the effect of training the two tasks simultaneously, multiple architectures are considered that vary the amount of shared weights which can be viewed in Fig. 3.2.

Each network can be viewed as a function M_f, where f represents the model weights, which maps a 300×300 pixel input image I to four parameter-space output images: $Q_f(I)$, $\Theta_f^{\sin}(I)$, $\Theta_f^{\cos}(I)$, $W_f(I)$ with an added auxiliary classification branch $A_f(I)$. Each image output represents a pixel map the same size as the input. The first four of these form the grasping output and come from one branch of the network, and the final auxiliary output A comes from the the classification branch. The pixel values of $Q_f(I)$ represent the probability of a successful grasp at each pixel position in I. The angle of the proposed grasp can be calculated pixel-wise by $\Theta(I) = \arctan(\frac{\Theta^{\sin}(I)}{\Theta^{\cos}(I)})/2$. The raw angle Θ of each grasp is in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$; it is represented as two unit vector components $\cos(2\Theta)$ and $\sin(2\Theta)$ in the network outputs as this aids learning by removing the discontinuity at the wrap-around point [154]. Finally, $W_f(I)$ is the required gripper width as a value in the range $[0, 1]$, which is scaled into a physical measurement for gripper width in the range of $[0, 150]$.

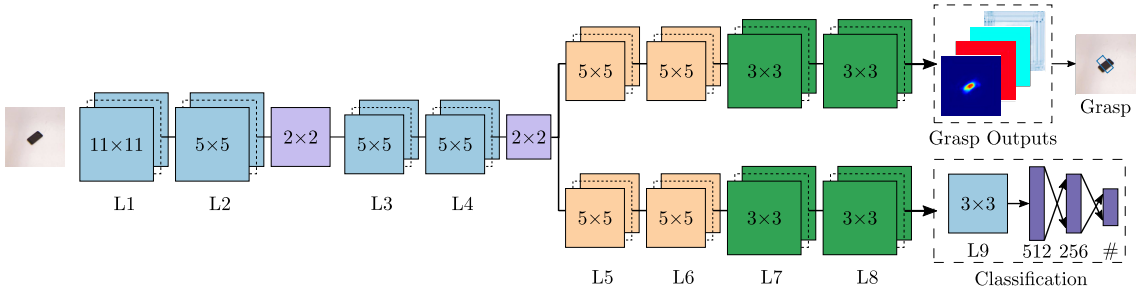
Each model architecture is a fully convolutional network consists of eight convolutional layers and two max pooling layers. The first half consists of a large convolutional layer with an 11×11 kernel size and a second convolutional layer with a smaller 5×5 kernel size, followed by a 2×2 max pooling layer, two further 5×5 convolutional layers, and a second 2×2 max pooling layer. Two dilated convolutional layers are then added from Yu and Koltun [159] which show increased performance in semantic segmentation tasks. The first



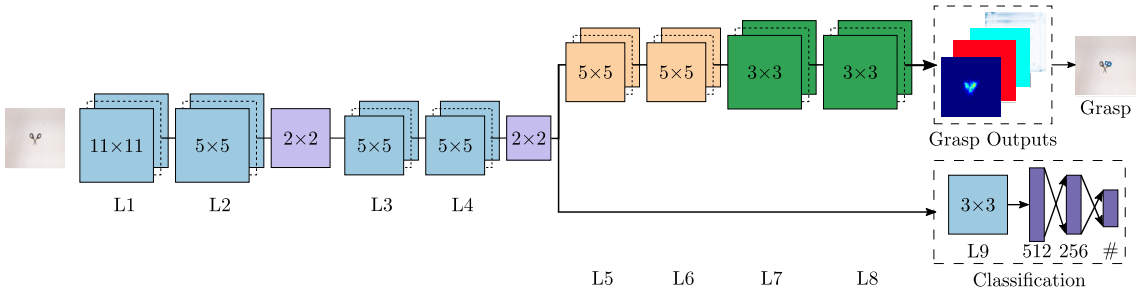
(a) MTG-CNN$_a$



(b) MTG-CNN$_b$



(c) MTG-CNN$_c$



(d) MTG-CNN$_d$

Figure 3.2: Variants of the fully CNN multi-task architectures with varying branch locations.

has a dilation of 2 and the second has a dilation of 4. Two 3×3 transpose convolutional layers are then added with a stride of 2 to increase the size of the output image to the original input image dimensions.

The way the architectures of these models differ is the amount of shared weights between the initial grasping task and the given auxiliary task. If the auxiliary task shares a large set of universal features with the initial grasping task then a greater amount of shared weights would improve both tasks. Therefore, the base MTG-CNN network, identified as MTG-CNN$_a$ (see Fig. 3.2a), is the most similar to the original GG-CNN2 network and contains the fewest parameters overall. Whereas, the GG-CNN2 network only produces a four parameter space output images $Q, \Theta^{\sin}, \Theta^{\cos}$ and W, corresponding to the components of a grasp from the original network, the MTG-CNN$_a$ features the additional classification A output module after the final layer of the model consisting of fully connected layers. The classification layer featuring nodes equal to the number of classes during training.

The MTG-CNN$_a$ features the greatest amount of shared weights between the proposed architectures, with only the final output layers allowed to distinguish between grasping and classification. For the other subsequent architectures, the number of outputs remain the same, but the number of parameters is increased to allow the network to learn features unique to both tasks.

For these networks, the initial layers are still shared but later layers are replicated to allow for specific features to be learned. Therefore, the first variation replicates the two transpose layers which is referred to as MTG-CNN$_b$ (Fig. 3.2b). The second variation called MTG-CNN$_c$ (Fig. 3.2c) replicates both the the dilation and transpose layers, leaving only the initial convolutional layers to learn shared features. A third variation is included that removes these transpose layers, acting similarly to an ablation study, referred to as MTG-CNN$_d$ (Fig. 3.2d). This was used as an experiment to showcase the effect that removing specific network regions dedicated to classification had on training.

To extract a proposed grasp rectangle from the network, we take the (x, y) coordinates of the maximum pixel value in Q as the centre of the grasp, the angle is then extracted from the corresponding pixel coordinate values from $\Theta^{\sin}$ and $\Theta^{\cos}$ according to the equation above, and finally the width of the gripper from W. For smoothness, before extracting the proposed grasp the outputs are filtered with a Gaussian kernel with a standard deviation of 5 pixels, as is done in [160]. For the classification task, the output is first passed through

a sigmoid and the max value is retrieved for the predicted grasp.

Cornell Grasping Dataset

In order to train the grasp map based generative networks, the Cornell dataset must first be converted from the rectangle-based representation [77], from Section 2.3.1, consisting of the gripper position centre, angle, and width (x, y, Θ, W), to the image based representation $\hat{G}$ for grasp map outputs from [5]. Only positive valid grasp labels from the dataset are used, from which, the centre third of each grasping rectangle corresponding to the central position of the gripper is taken as an image mask to form the training examples. This approach assumes that any other area is not the location of a valid grasp and forms the component grasp maps including the grasp quality Q, angle Θ and width W. The process is outlined in Fig. 3.3.

- Grasp Quality $\hat{Q}$: a binary mask representing each ground-truth positive grasp. Every pixel that falls within the centre third of a valid grasp rectangle is set to a value of 1 with all other values set to 0.

- Angle $\hat{\Theta}$: The corresponding angle for each grasping rectangle is also represented pixel-wise. For each pixel with a value of 1 in $\hat{Q}$, the value in Θ is set to the corresponding angle of the parallel gripper in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$ because the gripper rotation becomes symmetric around this point. To make learning easier for the network, the angle is further decomposed into the two components of a unit circle in the range $[-1, 1]$ as in Section 3.1.2, representing Θ^{sin} in the range of $[0, 1]$ and $\Theta^{\cos}$ in the range of $[-1, 1]$. Any discontinuities where the angle wraps around $\pm\frac{\pi}{2}$ radians if the raw angle was used as the parallel gripper was symmetrical around this point. The angle is then recalculated during testing by taking the value from $\sin(2\Theta^{\sin})$ and $\cos(2\Theta^{\cos})$ and recombining them via the equation:

$$\Theta = \arctan(\frac{\sin(2\Theta^{\sin})}{\cos(2\Theta^{\cos})})/2 \tag{3.7}$$

- Width $\hat{W}$: A pixel-wise representation of the required width for the parallel gripper. To allow for depth invariance, values are in the range of $[0, W_{max}]$ pixels where $W_{max} = 150$. This can be transformed into a physical measurement during deployment on a robotic arm using the depth camera parameters and measured depth from

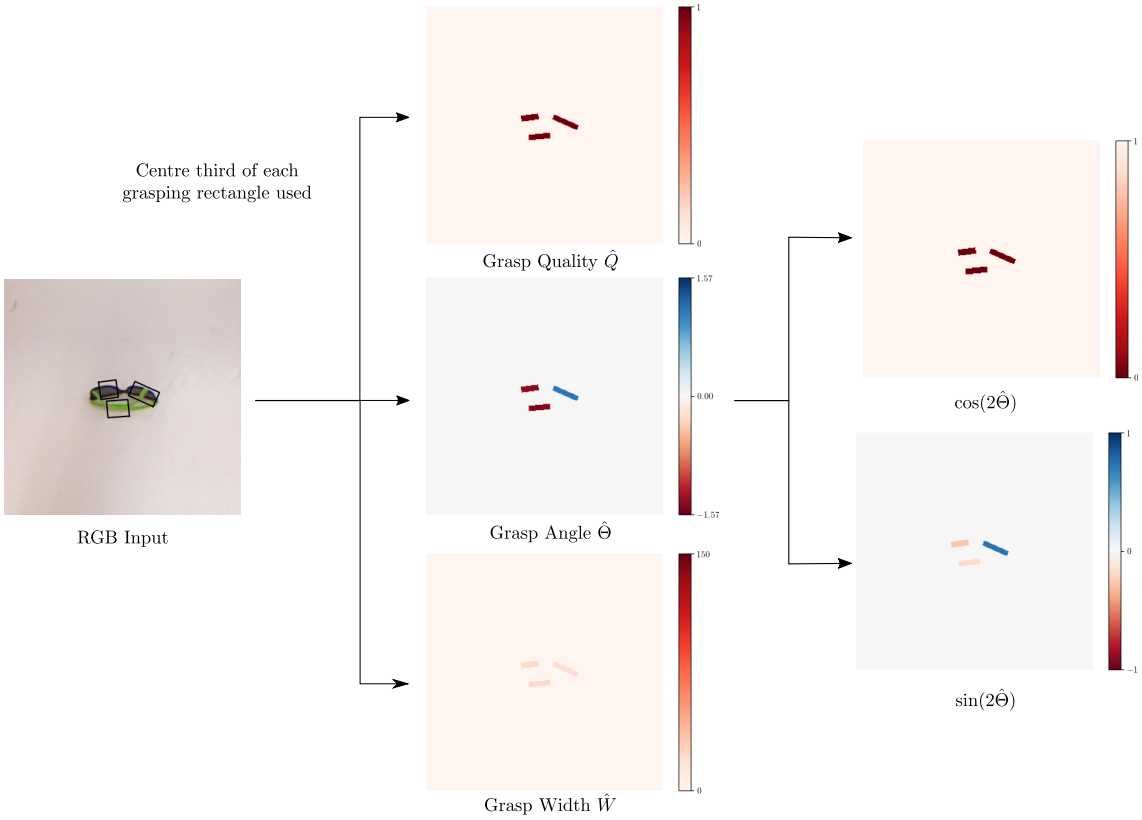


Figure 3.3: Annotated data from the Cornell dataset are transformed into grasp maps for training. The centre third of each grasp is transformed into a training image for grasp quality Q, grasp angle Θ, and grasp width W, with grasp angle further decomposed into sin & cos.

the gripper to the base of the arena. During training these values are scaled to the range $[0, 1]$ by multiplying by $\frac{1}{150}$.

Object classifications for the Cornell dataset were hand-labelled according to the two methods outlined in Section 3.1.1. Therefore, each object was classified and saved in the common COCO classification format [157]. Each object received a *specific* object ID as well as a *general* super-class that grouped objects with similar functions together to direct learning based on objects with similar grasps. Each object was also labelled with a bounding box which is included within the annotations, however this was not utilised in these experiments.

Performance Metrics

Performance of each trained model was measured according to both the proportion of successful grasps on the validation set and related classification performance. A successful grasp on the Cornell dataset is calculated according to the intersection over union (IoU)

metric from [42, 78] as used in [38, 129]. This is a quick offline metric that considers a proposed grasp as successful against two criteria:

- the predicted grasp rectangle and a corresponding ground truth grasp rectangle share an Intersection over Union (IoU) score of greater than 25%;
- the offset of the predicted grasp rectangle aligns within 30° with the corresponding ground truth grasp rectangle.

Object classification is considered successful when the predicted output was equivalent to the ground truth, however, as the main goal of the auxiliary branch is to improve grasp performance, the model with the highest IoU performance is reported alongside classification.

Loss Function

Due to the architecture of the MTG-CNN architecture, two separate loss functions were applied during training to grasping and classification separately. Learning to grasp was treated as a regression problem compared to the auxiliary classification problem. For each grasp output, a mean squared error (MSE) loss, also known as squared L2 norm, was applied to each predicted output separately against the ground truth resulting in the total grasp loss being calculated as:

$$\ell_{grasp} = \ell_{MSE}(Q, \hat{Q}) + \ell_{MSE}(\Theta^{\sin}, \hat{\Theta}^{\sin}) + \ell_{MSE}(\Theta^{\cos}, \hat{\Theta}^{\cos}) + \ell_{MSE}(W, \hat{W}), \tag{3.8}$$

For the auxiliary classification output, the output is first passed through a sigmoid applying a negative log likelihood loss (NLL), with x representing the predicted classes for a given batch and y representing the corresponding ground truth classes for the same batch:

$$\ell_{aux} = \ell_{NLL}(x, y), \tag{3.9}$$

Therefore, the total loss for the network is equal to the addition of these two terms:

$$L = \ell_{grasp} + \ell_{aux}, \tag{3.10}$$

However, the two error values produced from these loss functions can differ substantially as one is a classification task and the other is a regression. Therefore the units, in some

cases, can differ by orders of magnitude. An additional loss term is applied similar to a naïve weighted linear sum of losses for each individual task where λ is manually fine-tuned equal to 1 that forces the network to learn a preference for a task. This weighted linear sum is added to the loss function:

$$L = \lambda \ell_{grasp} + (1 - \lambda)\ell_{aux}, \tag{3.11}$$

This is compared to a more intelligent learned multi-task loss from Kendall *et al.* [107], which is a principled multi-task loss function that learns to maximise the Gaussian likelihood of the model based on task-dependant uncertainty. Effectively this learns to maximise the log likelihood with respect to the model parameters based on an observed noise parameter σ. This multi-task loss can be passed to the same optimiser used by the network to adjust learning rates between multiple outputs and used an example to show that multi-task performance was improved even across different modalities. In this instance, each task loss is weighted according to the dynamic weighting σ learned separately for each task such that the total loss becomes:

$$L = -\log \sigma_1^2(\ell_{grasp}) + -\log \sigma_2^2(\ell_{aux}). \tag{3.12}$$

Training

Due to the small size of the dataset, 90% of the original 885 images was kept for training and the last 10% was used for evaluating model performance in order to maintain a sufficient number of training examples. However, as this dataset was not originally designed with classification in mind, this culminated in a skewed dataset with an imbalanced set of objects. The distribution of objects in the *specific* training are shown in Fig. 3.4 with the test set displayed in Fig. 3.5. The corresponding *general* distribution of training labels are shown in Fig. 3.6 and the testing labels in Fig. 3.7.

As objects in the dataset were collected mainly from the convenience of object availability, rather than diversity, it was noted that there were an overabundance of some examples and sometimes only one instance of other objects. For example, during the first pass of object labelling it was noted that there was an overabundance of bottles within the dataset. Therefore, each instance of *bottle* was split further according to function (*pour bottle*, *squirt bottle*, and *spray bottle*) to reduce the chance of the network only learning to predict the most common object type. However, because only the final 10% of the dataset

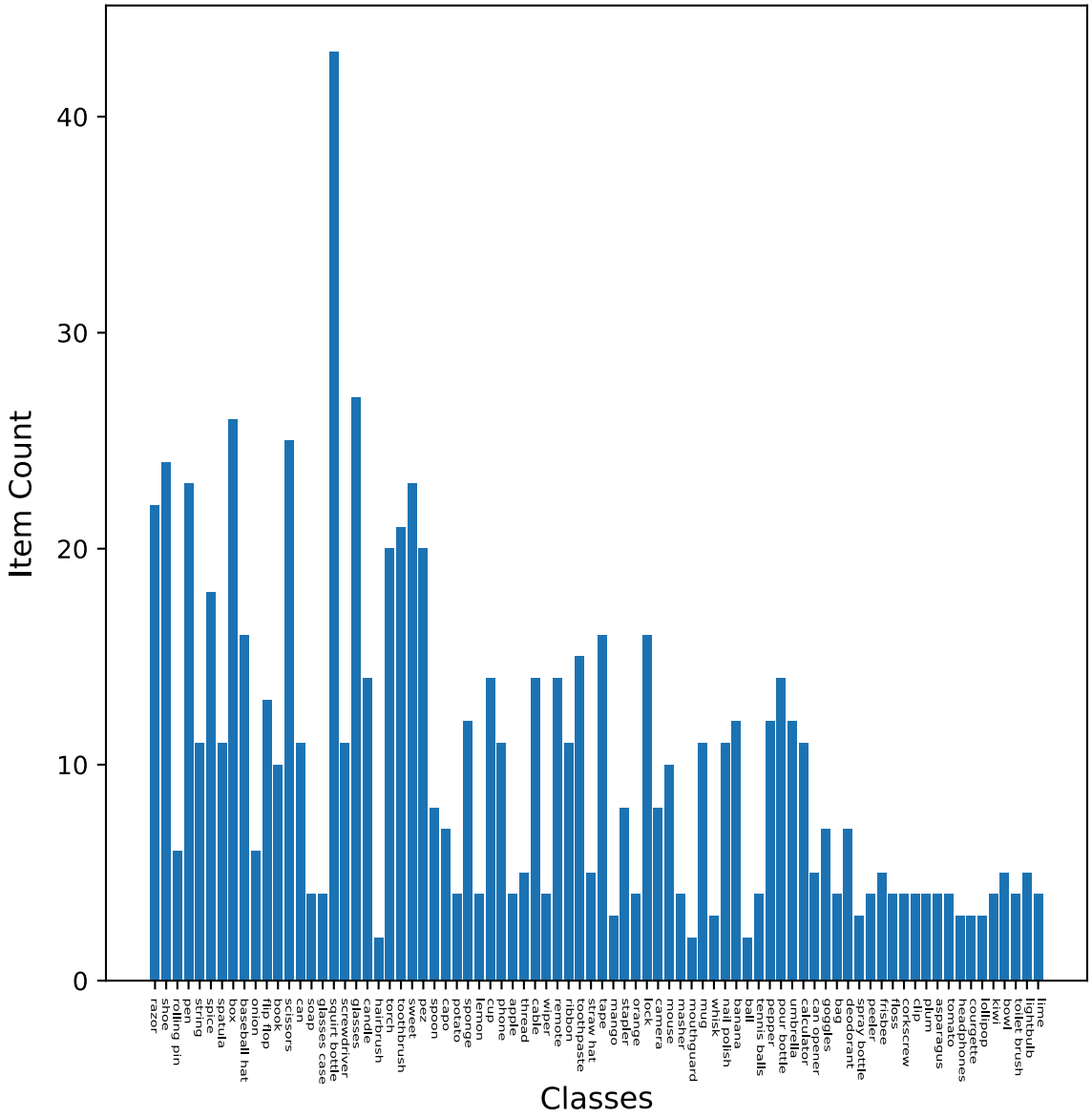


Figure 3.4: Distribution of training labels for *specific* object ground-truth labels.

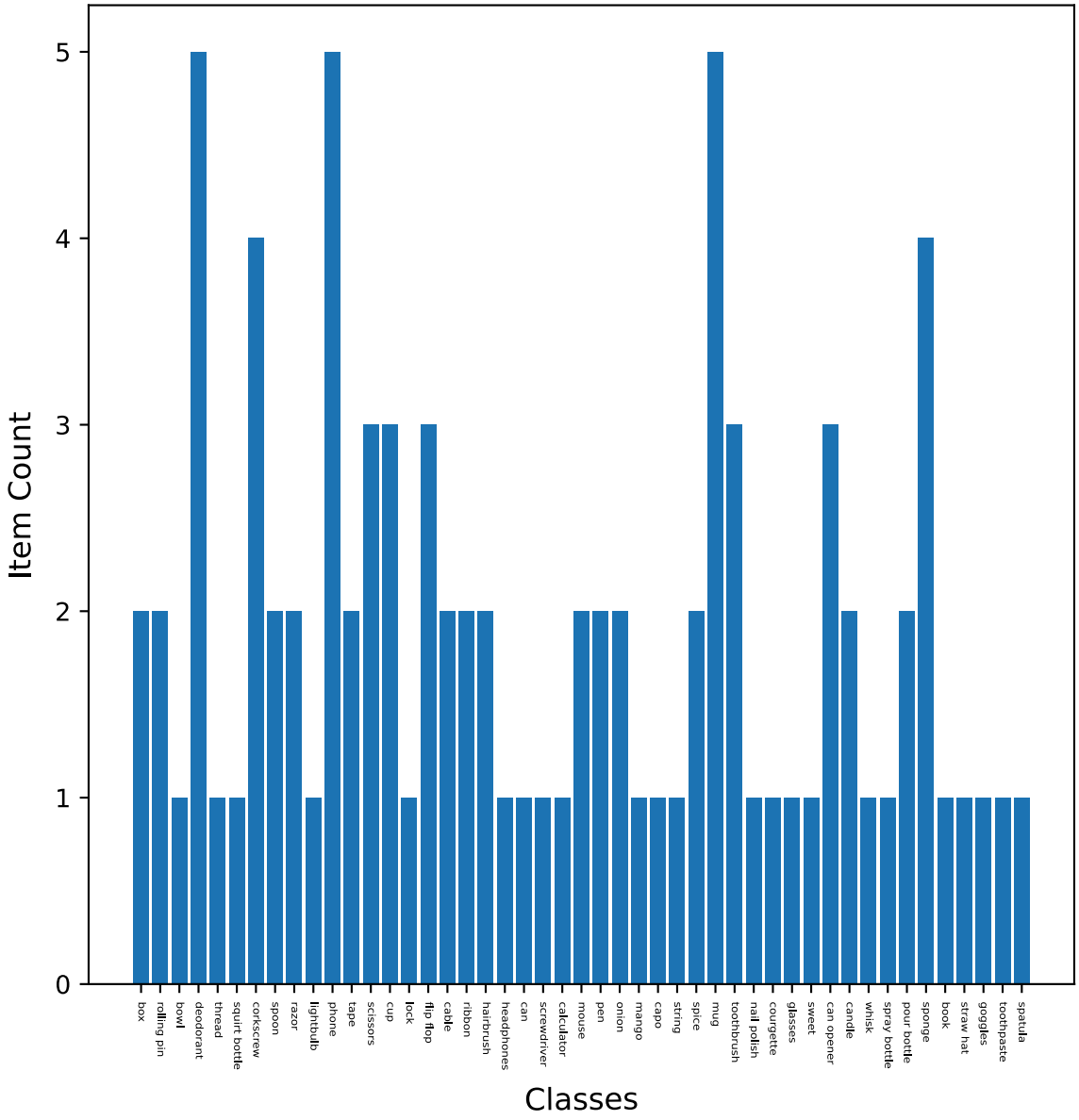


Figure 3.5: Distribution of testing labels for *specific* object ground-truth labels.

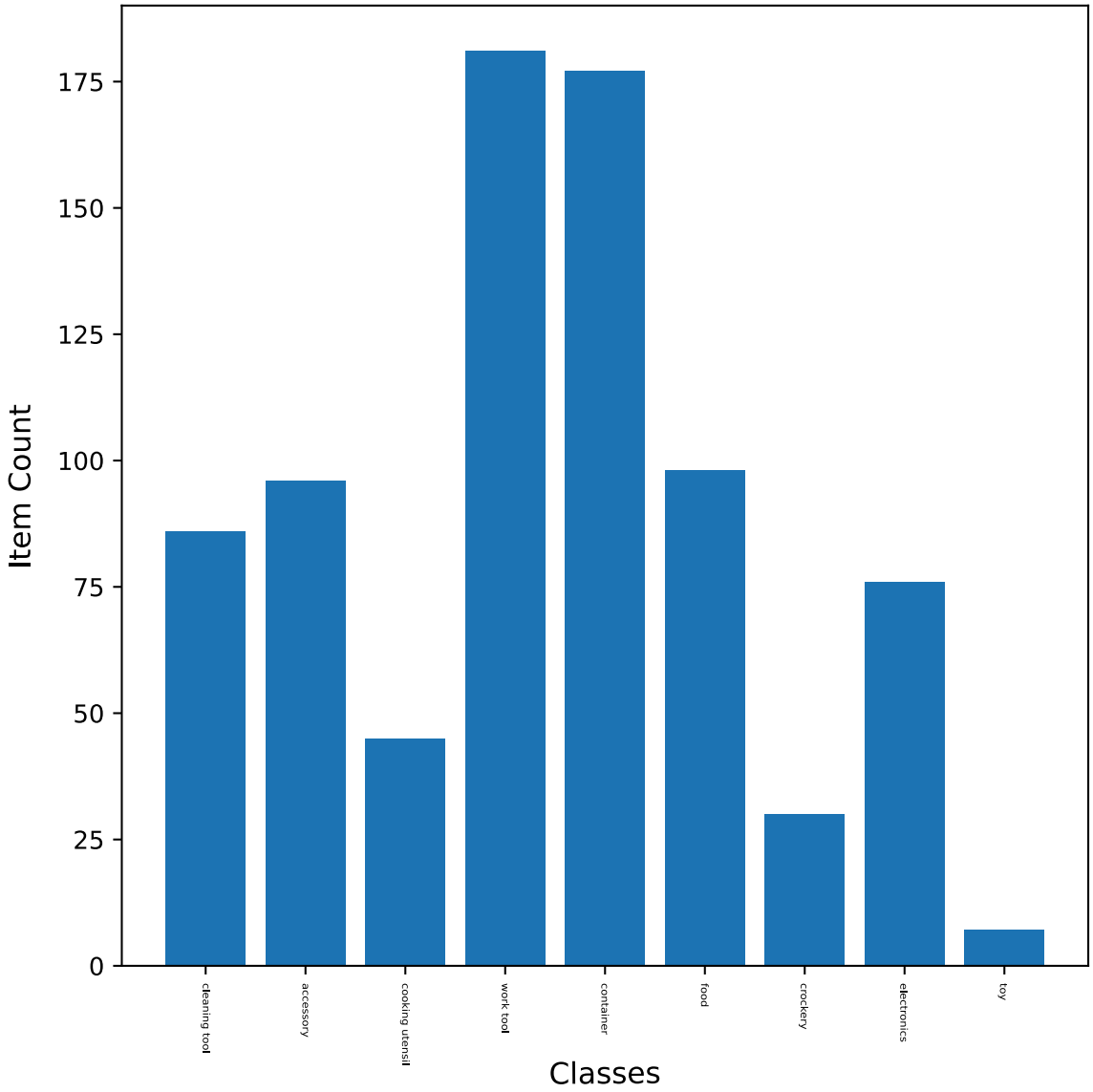


Figure 3.6: Distribution of training labels for *general* object ground-truth labels.

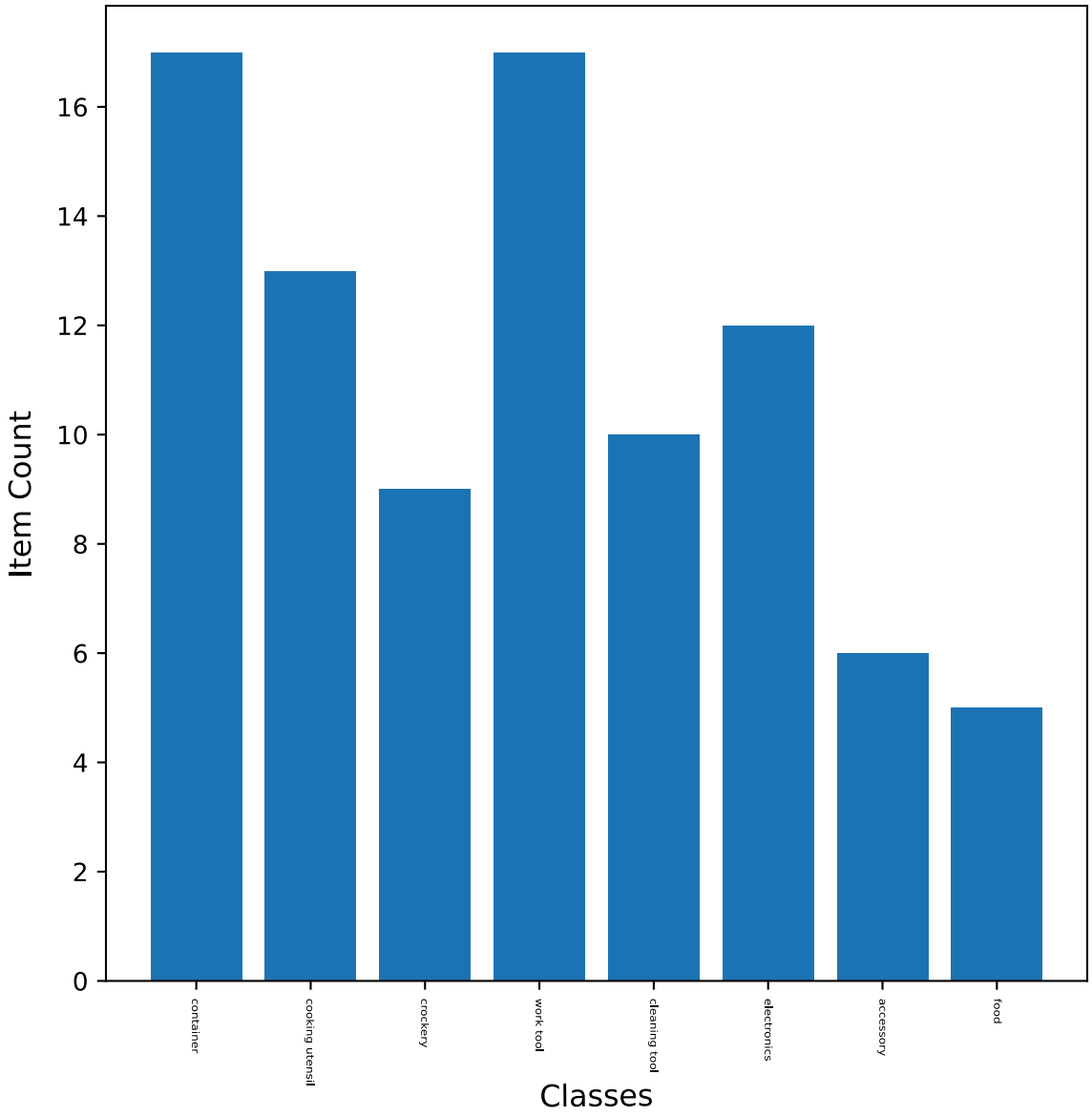


Figure 3.7: Distribution of testing labels for *general* object ground-truth labels.

was used in the evaluation set, this resulted in some categories missing from the test data which is more prominent in the *specific* object categorisation as there is some categories with only one instance within the training set. This created a sample where 48 categories where present in the *specific* test set. Whereas, 8 of the 9 object categories were present in the test set for the *general* object classification task.

Each network architecture was trained for a total of 40 epochs when trained with the weighted sum loss function using hand-tuned weights and the task uncertainty loss from [107]. The model with the highest grasping performance when trained with both types of auxiliary classification task was then tested on the final 10% of the CGD which formed the final results.

3.1.3 Evaluation

The performance of each network is evaluated against both the Intersection over Union (IoU) metric (see Section 3.1.2), as well as the classification results when trained to recognise the *specific* set or *general* set of object classifications. As the aim of these experiments was to analyse whether the grasping performance could be improved via the the introduction of the auxiliary classification task, the model with the best average performance between each task is tested, even though this may have not been the highest classification performance overall.

Results for the model performance on the test set when trained to recognise the *specific* object class are reported in Table 3.1, and the *general* object class results are reported in Table 3.2. In each case, individual architectures are trained using RGB colour input on both the grasping and classification tasks simultaneously. Furthermore, task specific loss functions are weighted due to the different scales used for the regression loss functions and classification function, as described in Section 3.1.2. This is applied individually to each model which differ by the number of parameters dedicated to learning each task individually, with MTG-CNN$_a$ featuring the greatest number of shared parameters and MTG-CNN$_c$ featuring the least. The effect of both the loss weighting (λ) and number of shared parameters are evaluated and compared to the base GG-CNN2 network [5] when trained to perform each task individually.

Table 3.1 shows the performance results for grasping with a corresponding *specific* object classification task. Overall, this shows mixed results for auxiliary classification performance. The first two sections show the baseline GG-CNN2 performance when trained

Table 3.1: Percentage of correct grasps according to the IoU metric and the *specific* object classification performance.

Model	**Input**	λ	**Performance (%)**	
			Grasp IoU	Specific
GG-CNN2	RGB	1.0	87.55	-
	D		89.16	-
	RGB-D		88.35	-
GG-CNN2	RGB	0.0	-	1.21
	D		-	1.21
	RGB-D		-	1.21
MTG-CNN_a	RGB	0.5	46.18	28.51
		0.6	90.76	1.21
		0.7	91.16	1.21
		0.8	50.60	**56.22**
		$-log\sigma^2$	87.15	42.17
MTG-CNN_b	RGB	0.5	67.47	31.73
		0.6	79.52	1.21
		0.7	75.90	6.82
		0.8	80.32	13.65
		$-log\sigma^2$	85.14	37.75
MTG-CNN_c	RGB	0.5	85.54	36.14
		0.6	87.95	1.21
		0.7	91.97	1.21
		0.8	80.32	33.33
		$-log\sigma^2$	89.16	30.12
MTG-CNN_d	RGB	0.5	90.76	1.21
		0.6	**92.77**	1.21
		0.7	88.35	1.21
		0.8	**92.77**	1.21
		$-log\sigma^2$	90.36	1.21

Table 3.2: Percentage of correct grasps according to the IoU metric and the *general* object classification performance.

Model	**Input**	λ	**Performance (%)**	
			Grasp IoU	Generic
GG-CNN2	RGB	1.0	87.55	-
	D		89.16	-
	RGB-D		88.35	-
GG-CNN2	RGB	0.0	-	20.08
	D		-	20.08
	RGB-D		-	20.08
MTG-CNN_a	RGB	0.5	90.76	20.08
		0.6	89.16	20.08
		0.7	89.56	20.08
		0.8	88.76	20.08
		$-log\sigma^2$	71.49	60.64
MTG-CNN_b	RGB	0.5	65.06	50.60
		0.6	79.92	36.55
		0.7	81.53	**58.23**
		0.8	82.73	55.42
		$-log\sigma^2$	83.13	57.43
MTG-CNN_c	RGB	0.5	89.96	20.08
		0.6	87.15	20.08
		0.7	80.32	50.60
		0.8	89.96	33.33
		$-log\sigma^2$	85.94	52.21
MTG-CNN_d	RGB	0.5	90.76	20.08
		0.6	87.95	20.08
		0.7	**90.36**	20.08
		0.8	88.76	20.08
		$-log\sigma^2$	**90.36**	20.08

using different input modalities on each task separately. Whereas grasping performance remains high, the network is unable to distinguish between objects correctly. In this instance, all three modalities default to predicting the object with the highest instance in the training set (*squirt bottle*). This remains a common theme across a number of MTG-CNN models. This is likely due to such a large imbalanced dataset where some classes dominate in prevalence. However, there are numerous cases where the networks learn to distinguish between classes whilst grasping performance remains relatively high.

Similarly Table 3.2, shows the same performance results when models are trained to recognise the *general* object classification instead. In this instance, the baseline GG-CNN2 network also learns to predict the class which features the most frequently in the training set (work tool), leading to poor overall performance. These classification results are generally higher than the *specific* class because there are fewer classes to distinguish between, however, grasping performance is generally lower as a result.

Effect of Model Architecture

With an increasingly earlier branch point, the number of model parameters, and therefore layers dedicated to learning specific task functions, also increases. However, in both the *specific* and *general* object classification tasks, whilst the grasping performance generally increases upwards, the classification results generally decrease.

The highest *specific* and *general* classification performance was reported in both cases by the MTG-CNN$_a$ model with a 56.22% and 60.64% success rate respectively. However, for this model, when classification performance is high, there is a large impact on the grasping performance, which only achieves a corresponding 50.60% and 71.49% grasp success rate. This is far lower than the highest grasping performance achieved by the network of 91.16%, but this only occurs with the traditional weighted sum when the classification performance collapses.

The MTG-CNN$_a$ features the highest number of shared parameters between grasping and classification which may reduce the ability of the network to distinguish between the overall performance of the two classes by preventing the final layer from learning task specific information. It is interesting to note, that in the cases where classification performance deteriorated, the highest grasping performance outperforms the GG-CNN2 baseline across all modalities.

The highest grasping performance overall however, was achieved by the MTG-CNN$_d$

model with a reported 92.77% grasping success rate. In all cases, this model was unable to successfully interpret classification specific tasks without the later layers dedicated especially to the task, instead favouring to only predict the highest frequency object. However, the inclusion of the auxiliary task, for both *specific* and *general* object classification, was enough to improve grasping performance when isolated to only the early layers of the model. Even the inclusion of the task uncertainty loss was unable to balance performance across the two tasks.

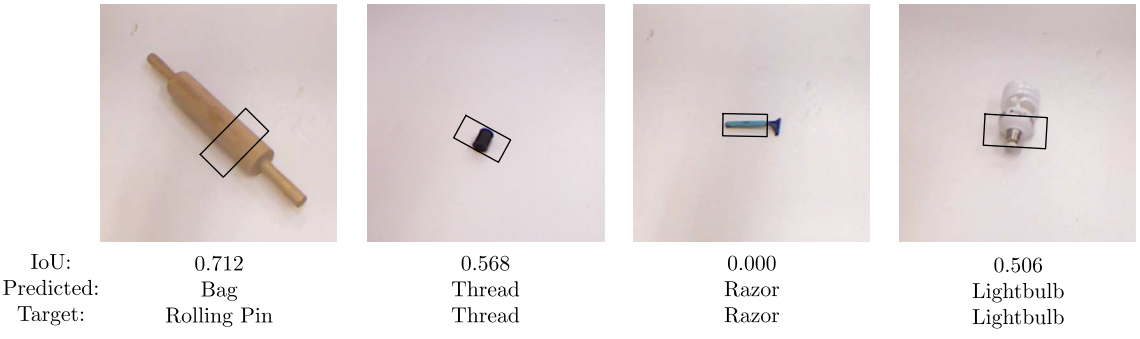
As the number of layers specific to each task increases, training generally becomes more stable at the cost of classification performance. Grasping performance remained high when classification performance improved for both the MTG-CNN$_b$ and MTG-CNN$_c$ models compared to the MTG-CNN$_a$ model, therefore suggesting that grasping and classification require the specific regions dedicated to each function similar to the human visual model unless the tasks are more related.

For greater performance across the two tasks, more specialised layers may therefore be required. Whilst the dilated and transpose layers provide a greater receptive field [159], similar to later cortical regions in both the ventral and dorsal streams [26, 161], they are designed to improve performance in semantic segmentation tasks. Instead of purely replicating the architecture for a new task, more specific layers suited to classification may provide improved results, such as transformer layers [162], that are better able to embed classification information closer to that of the ventral stream.

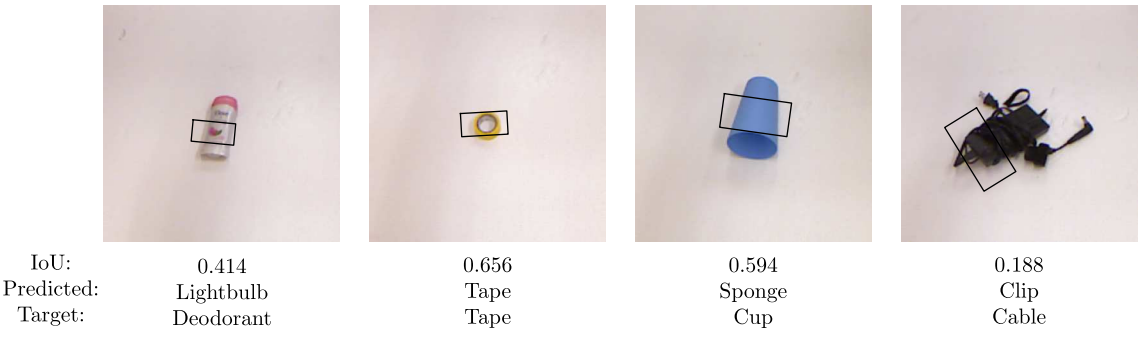
Effect of Classification Labels

Overall, there was no clear improvement to grasping performance when trained with either the *specific* or *general* auxiliary classification task. In both cases, the highest grasping performance was reported by networks that experience a catastrophic failure in learning the classification task. Although, this featured more in architectures without dedicated layers for object identification.

While classification performance was relatively higher for the *general* category, this is likely due to the reduced number of classes for the network to learn, making this an easier problem for the network to solve. This poor *general* classification performance may be because the 'tool' label, whilst having a distinct meaning and understanding to humans, the definition is broad that it may be difficult fully separate function from form. Here, it was used to specifically classify household hardware distinct from classes like kitchen



(a) MTG-CNN$_a$



(b) MTG-CNN$_b$

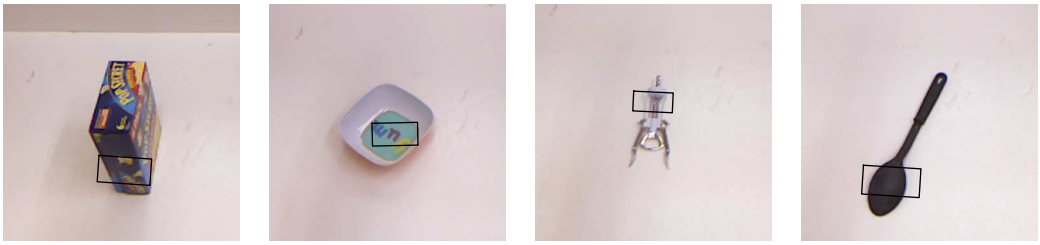
(c) MTG-CNN$_c$

Figure 3.8: Random examples of grasp outputs and corresponding predicted *specific* classification labels from MTG-CNN models..

utensils but features such a wide variety of functions that it becomes difficult to distinguish the tool-specific qualities. It may be unsuitable to include tool as a general classification type when humans are especially adapted to recognise tools [163] and feature specific cortical regions dedicated to tool use [155] due to their close relatedness with grasping as a function.

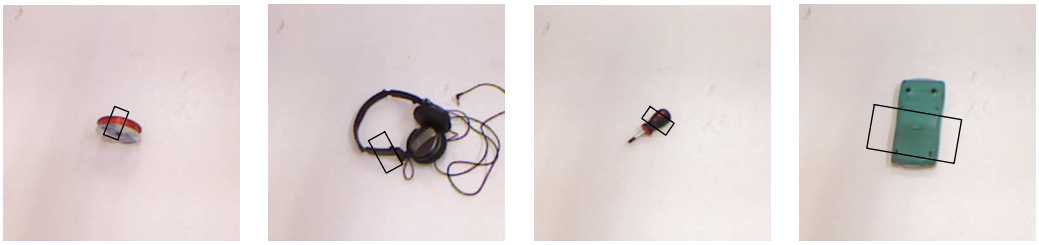
Effect of Loss Function

Generally, the weighted sum loss function tended to favour either grasping or the classification task without the ability to update loss weights during training. This resulted in a greater failure rate for classification overall, whereas all networks still demonstrated



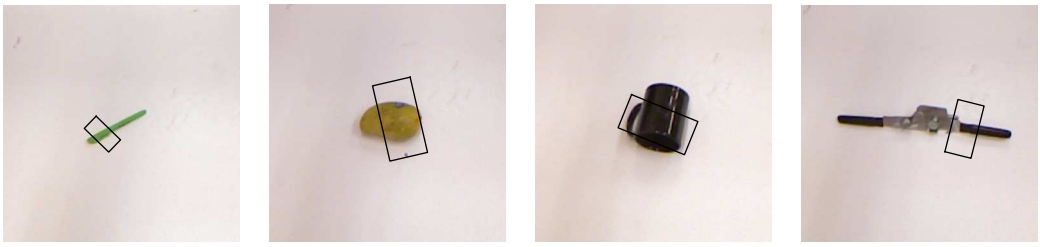
IoU:	0.476	0.04	0.316	0.660
Predicted:	Container	Crockery	Work Tool	Cooking Utensil
Target:	Container	Crockery	Kitchen Utensil	Cooking Utensil

(a) MTG-CNN$_a$



IoU:	0.552	0.000	0.573	0.494
Predicted:	Work Tool	Electronics	Work Tool	Container
Target:	Work Tool	Electronics	Work Tool	Electronics

(b) MTG-CNN$_b$



IoU:	0.479	0.603	0.541	0.467
Predicted:	Cleaning Tool	Container	Food	Cooking Utensil
Target:	Work Tool	Food	Crockery	Cooking Utensil

(c) MTG-CNN$_c$

Figure 3.9: Random examples of grasp outputs and corresponding predicted *general* classification labels from MTG-CNN models.

high performance on grasping. This suggests that a typical weighted sum is not enough to enable equal learning across the large difference in scales between the regression-based grasping and classification. On the other hand, the task uncertainty loss resulted in greater training stability between the two tasks, although, this did not always result in the best performance for either task for a given model. Instead, this method preferred to learn to heavily weight the grasping task during the first couple of epochs before reducing the weight disparity and learning the classification as a separate task.

It was noted during training that when using the log variable function, the loss scaling heavily skewed towards the classification task. This negatively impacted the ability to learn both tasks simultaneously suggesting that the classification task was significantly harder from such a small dataset. Training may benefit from multi-task learning featuring tasks trained with similar loss scales. Treating classification as a regression function, such as performing segmentation before classification may yield greater performance or training grasp classification based on intended function, similar to work from [36], that learned grasp configuration by classifying the semantic relation from one object to another. The Cornell dataset may be too small and inappropriate for such training examples.

3.2 Saliency and Depth Reconstruction

One of the main drawbacks of the classification auxiliary task was the requirement for extensive human labelling of grasping datasets, or the development of an entirely new dataset that provided improved balanced and integrated training data.

Therefore, the aim was then to consider data that would be readily available in existing grasping datasets to improve performance. This resulted in two choices: a saliency detection task; and a depth reconstruction task.

3.2.1 Background

Saliency refers to the selective attention towards an object that is visually distinct from the immediate surroundings. Koch & Ullman [164] was the first to note the relevance of saliency in human visual systems, and suggests the process of identifying salient objects is influenced by a number of factors, including how different an object is from surrounding information such as colour, motion, and depth.

In humans, saliency is crucial in guiding attention mechanisms which aid with object

grasping. For example, a large proportion of neurons in the posterior parietal cortex (PPC) within the dorsal area are dedicated to controlling attention as well as movement planning. This area is made up of three functionally distinct areas which are involved in higher order complex tasks such as spatial attention [165, 166], spatial navigation and decision making [167]. These distinct regions are: the lateral intraparietal area (LIP), specialised for saccadic eye movements; the parietal reach region (PRR), focused on arm reaching; and the anterior parietal area (AIP), involved in grasp planning and contains cells that determine the size and shape of an object [168, 169]. The information processed by these areas directly controls feedback mechanisms to areas that control attention and eye movements, like the medial superior temporal area (MST), an area that controls eye vergence [170], and the superior colliculus, an area that controls eye saccades [171]. This is functionally unique from the ventral stream as it contains direct feedback mechanisms to control eye movements for the direct influence of attention towards objects for updating arm reaching movements in space [169]. Computing saliency within a scene could therefore help improve robotic grasping systems by directing attention towards the objects easiest to grasp, i.e. non-occluded, visually distinct, or larger objects in a scene.

The tracking of salient objects was first shown to be successful in computer vision by Itti & Koch [172]. This was calculated by a model that replicated processes similar to the receptive fields present within the mammalian visual pathway, focusing on centre-surround contrast features and tracking colour, orientation, and intensity feature maps. Regional contrast in terms of global and local schemes have since been used frequently in conventional object detection and are proven to be efficient and effective in simple scenes with a single object. More recent studies introduced the binary segmentation for salient objects using conditional random fields (CRFs) [173] to capture global, regional, and local features. This reduced the task of salient object detection (SOD) to a binary labelling problem. Fortunately, saliency using binary labels can easily be mimicked within data using object masks, and is included in common datasets such as the Cornell [78] and Jacquard Grasping Datasets [129]. Therefore, there is access to a large set of data for training multi-task networks.

In a similar manner, a second *depth reconstruction* task is chosen which aims to provide the same benefits as the concurrent saliency task. As depth information is also provided alongside grasping datasets and achieves improved performance compared to RGB information alone, this is also considered as an auxiliary task to reintroduce information back

into the RGB scene that helps improve grasp performance. Depth information is likely to contribute contour information and distinguish an object from the surrounding environment, like saliency information, and takes advantage of similar processes such as CRFs when reconstructing depth using neural networks [174].

3.2.2 Methodology

The following section details the changes made to the methodology after the classification experiments. These methods correspond to the saliency detection and depth reconstruction experiments.

Jacquard Dataset

As depth and saliency information are more readily available compared to detailed classification labelling, we take this opportunity to switch to a more recent dataset with more concentrated and accurate grasp information.

Instead of the previous Cornell Grasping Dataset [78], models were trained with the much larger Jacquard Grasping Dataset (JGD) [129]. This is a simulated dataset containing a much larger 54,485 images of over 11,000 different objects on uniform white backgrounds. The images are annotated with over 1.1 million successful grasps also represented as grasp centre, angle and gripper width.

All grasps were generated within a simulated physics environment which attempted grasps at many positions, angles and widths; unsuccessful grasps and highly similar grasps were not recorded.

Every object in the dataset has four viewing angles with each viewpoint consisting of a single RGB image as well as a perfect depth image recorded from the simulated data and a generated stereo depth image.

Labelled grasping rectangles for each object from the dataset were transformed in the same way as the CGD into corresponding images to train each aspect of the learned grasp. Ground truth grasps from the dataset are labelled as $\hat{G}$ as before, whereas grasps generated from the network are labelled with G. For each image with a corresponding object I, a grasp quality image $\hat{Q}$, grasp angle $\hat{\Theta}$, and gripper width $\hat{W}$ are generated (see Section 3.1.2).

In order to train the grasping branch of each model, each ground truth grasp from the dataset was once again decomposed into the corresponding components which formed the

training images. For $\hat{Q}$ the pixel values are 1 if the pixel falls within the centre third of the successful grasping rectangle for I and 0 otherwise. Values of $\hat{\Theta}^{\cos}, \hat{\Theta}^{\sin}$ and $\hat{W}$ are set to according to the angle and width of a corresponding successful grasp centred at that pixel position, where $\hat{Q} = 1$ and are set to 0 elsewhere. $\hat{W}$ is the required gripper width as a value in the range $[0, 1]$, which can be scaled into a physical measurement for gripper width.

The angle of the grasp is in the range $[-\frac{\pi}{2}, \frac{\pi}{2}]$, this is due to the symmetrical nature of the gripper jaw so that grasps only need to be considered between these angles. However, the angle is not learned directly during training and is decomposed into two unit vectors, Θ^{sin} and Θ^{cos}, in order to improve the efficiency of training. This removes any discontinuities where the angle wraps around $\pm\frac{\pi}{2}$, and provides unique values within $\Theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ [154]. The angle of the proposed grasp can be calculated pixelwise in post-processing by $\Theta = \arctan(\frac{\sin(2\Theta^{sin})}{\cos(2\Theta^{cos})})/2$.

To train the saliency branch of the multi-task model, the included image mask that corresponded with each viewpoint for an object was used. This binary image mask consisted of a 1 where the object was placed in the scene and 0 everywhere else.

Alternatively, when training the depth reconstruction, only the true simulated depth image from the dataset was used to train the models. This image was a true depth representation of the scene with each pixel representing a measurement between the lens of the simulated camera and the base of the platform in the scene.

Training

For training and testing, the dataset is split according to the suggested methods provided alongside the dataset [129]: a 5-fold cross validation was performed where the data was split randomly. 80% of the data was reserved for training, whilst the remaining 20% of the data was split evenly between a validation and test set. All models are trained using the same 300×300 3-channel RGB input image and colour input data was normalised to the range of $[0, 1]$ before subtracting the image mean to centre the data at 0.

Data augmentation was applied to the dataset to artificially increase the amount of training data by random cropping and zooming on the images. Input data is cropped from its original size, resized, and normalised before being processed by the network to match the training data used and depth data is inpainted according to [175]. However, during testing augmentation was not applied and all images were presented in their original format

to keep testing performance consistent between runs. All models were trained using the Adam optimiser [176].

All models were trained for 40 epochs and the average overall performance is reported on the remaining 10% of data in the test set using the best performing model over the 40 epochs of training.

Performance Metrics

Grasping performance was once again measured using the IoU offline metric (see Section 3.1.2: a grasp is considered successful if the predicted grasp rectangle shares a greater than 25% IoU and aligns within 30° with the ground truth grasping rectangle. This measure provides an efficient offline metric for testing performance and was used in both the JGD [129] and CGD [78], the values presented in this section are from the Jacquard dataset [129].

The mean absolute error (MAE) is also reported to assess performance of both auxiliary tasks. In this case, a value closer to 0 indicates a closer predicted value to the observed value, and it can therefore be inferred that the network is performing better at reconstructing the saliency or depth image.

Loss Function

In addition to the grasping loss used in the previous experiments, the auxiliary loss from the previous loss function is modified to reflect the change in objective. Instead of applying a negative log likelihood (NLL) classification loss, the auxiliary term is replaced with an image-based regression loss to keep the loss between tasks scaled to a similar range. Both the saliency and depth reconstruction tasks are first trained with a typical MSE loss as with the grasping loss. This transforms the loss from Eq. 3.10 into:

$$\ell_{aux} = \ell_{MSE}(A, \hat{A}), \tag{3.13}$$

where A represents the given auxiliary task output and $\hat{A}$ refers to to corresponding ground truth for either salient object detection A_s or depth reconstruction A_d.

Alternatively, further task specific losses are applied that has previously shown to improve salient object detection performance. The commonly used binary cross entropy loss (BCEL) for SOD was used and the more recent consistency enhanced loss (CEL)

from [177] was applied to improve performance, as this was shown to further improve saliency detection compared to standard losses.

Here, BCEL is an element-wise loss function commonly used for SOD where the final prediction is calculated as:

$$\ell_{aux} = \ell_{BCEL} = \sum_{p,g} -[g \log p + (1-g) \log(1-p)], \tag{3.14}$$

where p represents the probability of a pixel belonging to a salient region in the range $0 \leq p \leq 1$, and g representing the ground truth. $\log(\cdot)$ is an element-wise operation. However, this loss function typically results in a saliency output images which does not consider the inter-pixel relationships across batches [177]. Therefore the CEL loss is added as an extra loss term to improve SOD. This can be written as:

$$\ell_{CEL} = \frac{|FP + FN|}{|FP + 2TP + FN|} = \frac{\sum(p - pg) + \sum(g - pg)}{\sum p + \sum g}, \tag{3.15}$$

where TP, FP, and FN represent true positive, false-positive, and false negative and $|\cdot|$ computes the area. $FP + FN$ indicated the difference set in union and intersection of the predicted foreground region and ground-truth one, while $FP + 2TP + FN$ represents the sum of this union set and intersection. This loss reinforces a global understanding of saliency whilst BCEL focuses on individual pixel values resulting in sharper boundaries in salient predictions.

The final CEL loss is therefore used in addition to the BCEL loss resulting in the final auxiliary loss:

$$\ell_{aux} = \ell_{BCEL}(A, \hat{A}) + \lambda \ell_{CEL}(A, \hat{A}), \tag{3.16}$$

with λ also representing a contribution weight between the two losses, however this is set to 1 for simplicity.

The generative grasp loss remains the same as previously used in Eq. 3.8 and therefore the total loss remains as:

$$L = \ell_{grasp} + \ell_{aux}, \tag{3.17}$$

with the auxiliary loss being represented by MSE, BCEL, or CEL loss for ℓ_{aux} during saliency estimation or just MSE for depth reconstruction.

The multi-task specific loss from [107] used in the classification experiments (represented as $-\log$) is also reapplied to this set of experiments as a learned weighting function

between the two branches of the network (see Section 3.1.2).

Multi-task Network Architecture

For the following set of experiments, a 5-fold cross validation was used to test the arge quantity of data only a single multi-task network was used to compare with the original GG-CNN2 network [5]. As the networks in the previous experiments performed better with extra layers dedicated to the auxiliary task only the MTG-CNN$_c$ network was used for comparison.

One branch is used for learning the grasp outputs, as in GG-CNN, the other branch for an auxiliary learning task, shown as depth reconstruction in Fig. 3.2c. The shared first half consists of two convolutional layers, followed by a max pooling layer and two further convolutional layers and a second max pooling layer. The two branches in the second half of the network are identical and consist of two dilation layers and two convolutional transpose layers, see Fig. 3.2. All convolutional layers have 16 filters except for the dilated convolutional layers which have 32 filters.

3.2.3 Evaluation

Each model was trained using a 5-fold cross validation on 80% of the Jacquard dataset with 10% of the data reserved as a validation set. Each model was then tested on the same unseen 10% of data as a fair comparison. Performance was measured using the intersection over union (IoU) method for grasping as previously used in the classification task alongside the mean absolute error (MAE) for assessing the quality of reconstruction during the auxiliary task (from Section 3.2.2), the results of which are presented in Table. 3.3. A higher IoU performance represents more successful grasps on the test set whereas a lower MAE indicates better auxiliary task reconstruction.

According to the results shown in Table 3.3, there is a large benefit to training a grasping neural network with a related auxiliary task that provides relevant task information using RGB input data. There is a general increase in grasping success when comparing the base GG-CNN2 network when trained with RGB input, compared to both versions of the MTG-CNN network trained with either a salient object detection task and a depth auxiliary task. The base GG-CNN2 network achieved an average grasp success of 72.04% whereas the best performing MTG-CNN models achieved 78.61% when trained with salient object detection (SOD) and a greater 79.50% grasp success rate when trained to perform

Table 3.3: Mean percentage of correct grasps evaluated on the Jacquard grasping dataset. Arrows indicate direction of better performance.

Model	**Input**	**Aux.Task**	**Aux.Loss**	**Grasps %** (↑)	**MAE** (↓)
GG-CNN2	RGB			72.04 ± 3.44	-
	Depth			82.26 ± 3.76	-
	RGB-D			83.11 ± 1.88	-
MTG-CNN	RGB	Saliency	MSE	78.61 ± 0.52	0.011
			BCEL	78.48 ± 0.97	0.007
			CEL	76.63 ± 2.24	0.005
			MSE$/-\log$	76.77 ± 1.37	0.009
			BCEL$/-\log$	78.12 ± 1.00	0.008
			CEL$/-\log$	77.63 ± 0.84	0.005
MTG-CNN	RGB	Depth	MSE	79.50 ± 0.65	0.022
			MSE$/-\log$	78.29 ± 1.76	0.014

concurrent depth reconstruction.

Whilst the highest grasping performance was still reported using the more information rich RGB-D input with a 83.11% grasp success rate, training with an auxiliary related task shows large improvements when depth information is removed as an input. Reintroducing more information using multi-task learning to the scene shows the network can learn to recover some of the lost information with depth reconstruction slightly outperforming saliency detection overall.

The MTG-CNN was able to achieve high reconstruction performance across both auxiliary tasks and ranges of loss functions. However, slight improvements to auxiliary task reconstruction did not equate to significant improvements in grasp performance. Both MTG-CNN models performed best at grasping when using the same mean squared error (MSE) loss as the auxiliary loss function, however this was not equivalent to the best auxiliary task performance overall. Whilst the highest grasping performance for the MTG-CNN network with SOD was achieved using the MSE loss at 78.61%, and a corresponding average MAE of 0.011, training with the superior consistency enhanced loss (CEL) resulted in the most accurate average mask reconstruction (0.005), this actually resulted in a decreased overall 76.63% grasping success rate. This suggests there is a limit to the effect that concurrent SOD understanding can do to improve grasp performance, as more resources dedicated to improving the auxiliary task begins to negatively impact grasp performance.

This is further backed by evidence that the multi-task loss ($-log$) from [107] had limited impact on increasing overall performance across both saliency and depth reconstruction,

whereas it was better able to weight the differences in function in the previous classification tasks. Whilst typically slightly improving auxiliary performance, there was no significant increase in grasping performance. Therefore, this suggests greater relevancy between the saliency and depth branches to the original grasping task.

Examples of corresponding grasp proposals and auxiliary outputs from the best performing MTG-CNN models are shown in Fig. 3.10 for models trained to perform SOD. According to the IoU grasp metric, a grasp is successful if there is a greater than 25% overlap of the grasp proposal and within 30° of the ground truth. Therefore, all grasps with an IoU above 0.25 in these figures are reported as successful.

From Fig. 3.10, the saliency branch of the network interacts with grasp proposals as the centre for a grasp is typically centred around the most salient parts of the object, avoiding areas that it calculates as less salient within the scene. This may be related to aspects such as ease of grasping or may be related to aspects that help achieve a grasp such as estimating the centre of weight distributions. As the object is individually present within the scene, saliency detection may be trivial for a network to calculate in comparison to more complex scenes, however introducing salient information may provide a useful heuristic within RGB scenes for grasping.

Further changes to the SOD loss function however, did not result in significant improvements to grasping, despite reporting slight improvements to the saliency auxiliary branch of the MTG-CNN model. A comparison of grasps and SOD outputs are compared in Fig. 3.11 between the MSE, BCE, CEL losses. This shows that general improvements to the auxiliary loss function have a limit if there is no further significant information to be gained from improvemed recognition of salient objects within the scene. However, each network still prefers to grasp the most salient point of the object according to the perceived saliency output.

On the other hand, models trained to perform depth reconstruction achieve slightly better performance on the Jacquard test set compared to those trained to perform saliency detection and the base GG-CNN2 network trained on RGB. Fig. 3.12 shows examples of grasp and depth reconstruction outputs of such models. Both depth reconstruction models start to achieve results closer to that of using depth as an input. As it is clear that depth information is crucial for understanding good grasp placement, learning to reconstruct this information shows better understanding of grasp placement compared to the base model.

Unfortunately, from the depth reconstruction examples in Fig. 3.12, there is evidence

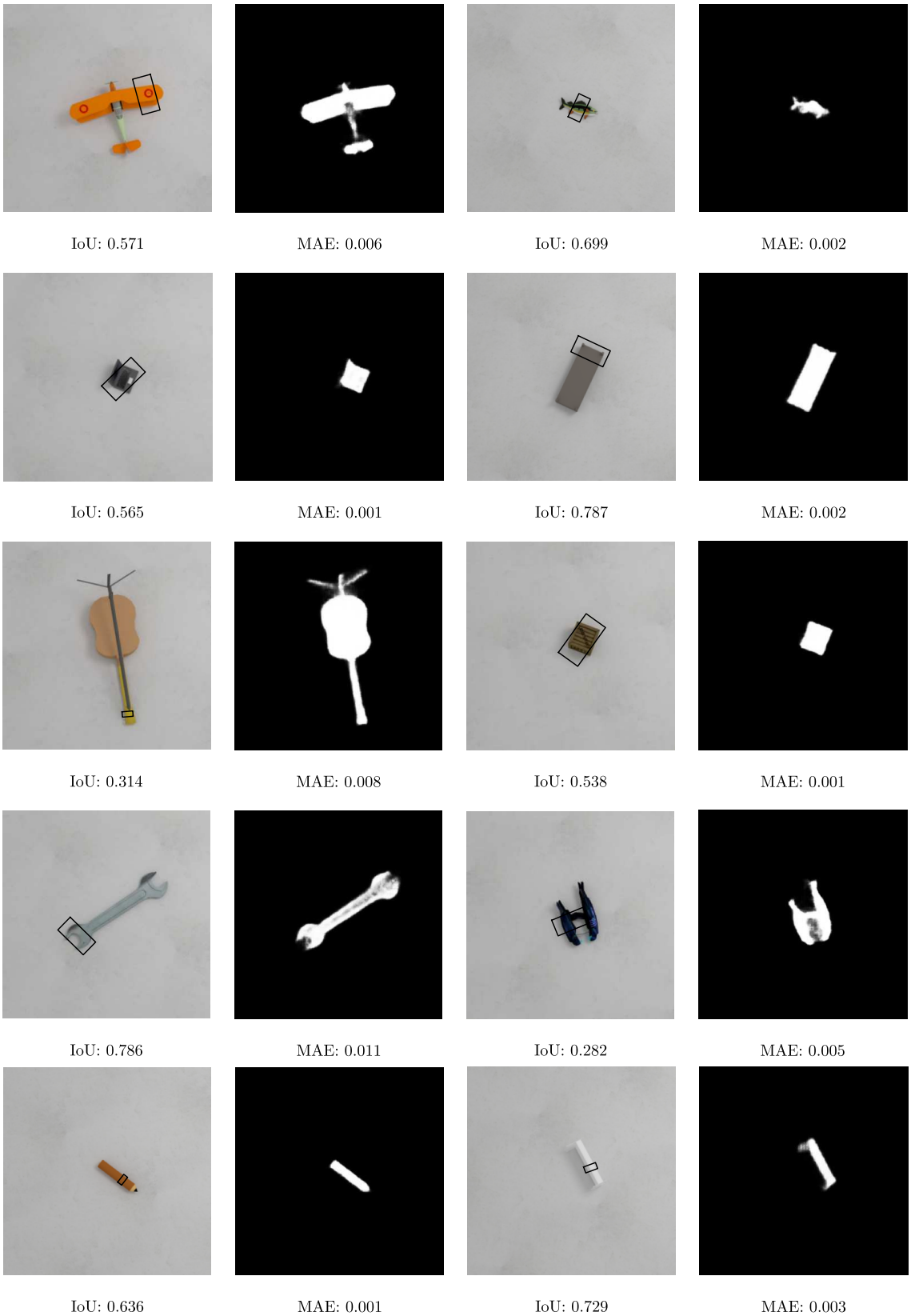


Figure 3.10: Random examples of proposed grasps from the MTG-CNN trained to perform a concurrent SOD task.

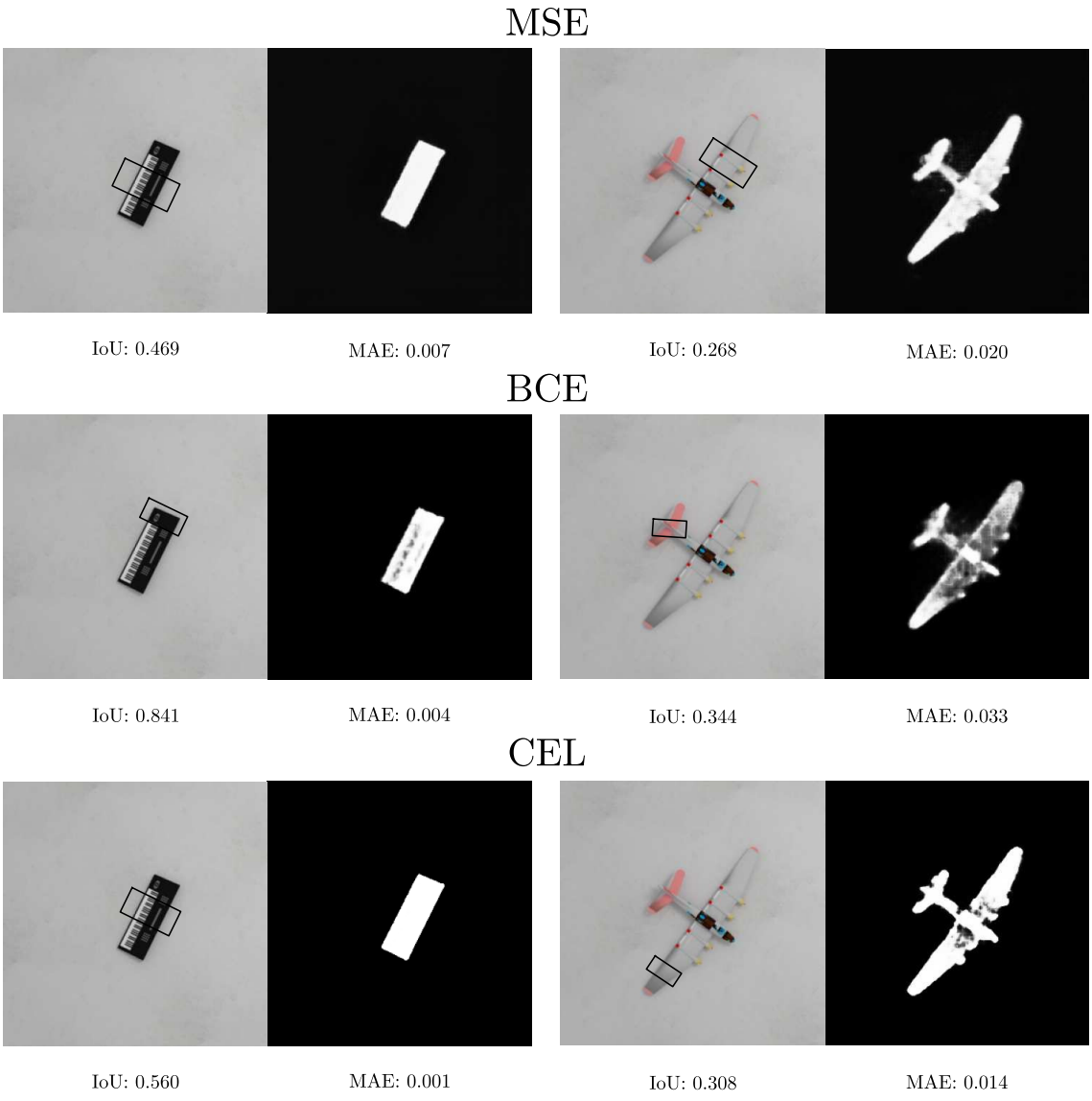


Figure 3.11: Comparison of model outputs between trained loss functions to perform salient object detection.

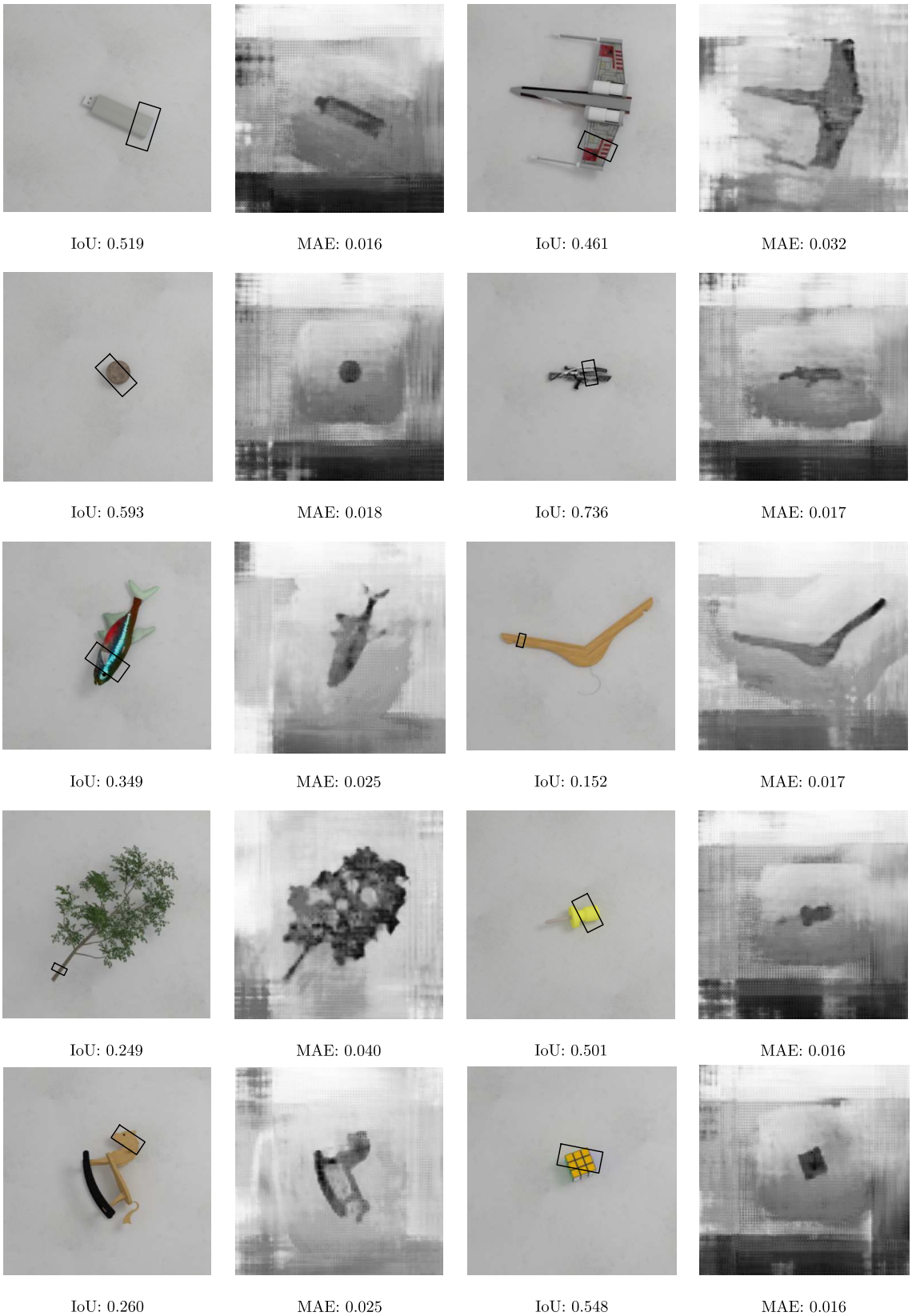


Figure 3.12: Random examples of proposed grasps from the MTG-CNN trained with an auxiliary depth reconstruction task.

of a boxing artefact present that was likely attributable to compression of image files during storage. However, the network appears to ignore these artefacts around the object suggesting that understanding the difference in depth between the object and surrounding area is important to grasping.

Both of these tasks show that reintroducing relevant information to the RGB scene can greatly improve grasping performance through multi-task learning. Furthermore, neither task significantly impacts inference time despite the increase in the number of parameters, and can therefore still be used in closed-loop robotic grasping as in [47].

3.3 Summary

The aim of this chapter was to investigate whether utilising a multi-task architecture can improve performance of a generative grasping model when trained using only RGB images. Three auxiliary tasks were considered to help provide more information with the intention of improving grasp understanding; classification of objects; salient object detection; and depth reconstruction.

Experimentation suggests that simultaneous end-to-end training of grasping and classification may be insufficient to improve grasping performance. The drastic difference in loss scale between the regression-based grasping task and the classification scale may be too large to properly learn either task to a high standard. The highest reported grasping results (92.77%) on the Cornell test set was reported by models that were unable to learn to distinguish between classes correctly, although this was still higher than the baseline model (87.55% to 89.16%). Whereas, models that were able to learn properties to correctly classify a set of objects, this information tended to have a negative impact on grasp performance and did not significantly improve performance compared to the base model.

With the introduction of a dynamic weighted loss function, such as the task uncertainty loss from [107], shows that the network actually benefits better from learning the two tasks sequentially. Therefore, whole object classification may not be the best method of grasp understanding. Grasp performance may be further improved from affordance classification rather than general object classification [36].

Varying the amount of shared weights in the network between the two tasks did result in changes to performance on both tasks. Increasing the amount of layers solely dedicated to each task generally improved training stability, allowing the network to learn task specific features whilst also forcing the network to learn shared inductions about both

tasks.

As a result, training multi-task networks to learn auxiliary saliency detection and depth reconstruction as alternative tasks resulted in clear improvements to generating accurate grasp proposals on the Jacquard test set. This is likely due to allowing for training on larger datasets without the need for hand classifying data from human experts and sharing closer features to the initial grasping task.

Salient object detection (78.61% ± 0.52) and depth reconstruction (79.50% ± 0.65) both improved grasp performance compared to the base network trained with RGB data (72.04% ± 3.44), nearing performance of using true depth data as an input (82.26% ± 3.76). However, this suffers from deteriorating performance, when attempting to further improve auxiliary task performance as this diverts attention away from grasp understanding. Saliency information helps isolate an object from background features whereas depth reconstruction helps reintroduce object specific information, such as curvature, which can improve grasping.

CHAPTER 4

Optimising Generative Grasping Models using Positional Loss

Of the two generative convolutional models introduced in the previous chapter, the generative grasping convolution neural network (GG-CNN) and the multi-task grasping convolutional neural network (MTG-CNN), both architectures are able to quickly predict a grasp proposal for each pixel in a scene by outputting the components of the grasp separately in the form of *grasp maps*. Utilising this method of grasp proposal is advantageous, in that, the model only requires a single forward pass to propose highly accurate grasps at multiple points on an object, resulting in faster operating times, compared to alternative discriminative models of grasping [4, 16], and that this method can easily generalise to unseen objects with a high success rate [5, 77].

Generating grasp proposals via the implementation of grasp maps allows the network to predict all available grasps in a scene by decomposing a grasp into multiple dedicated network outputs, including a grasp quality score that represents the probability of a successful grasp taking place on an object alongside the corresponding grasp map for a gripper angle and width value. This allows for a network capable of producing grasps with an inference time quick enough for the online control of a robot using visual servoing [49].

However, during training, it was noted that equal weighting was attributed to each output. We argue that this is unhelpful, to learning as a large proportion of the total computational loss generated using the standard loss functions, is dedicated to learning irrelevant components of a scene instead of the grasping task. This leads to long training

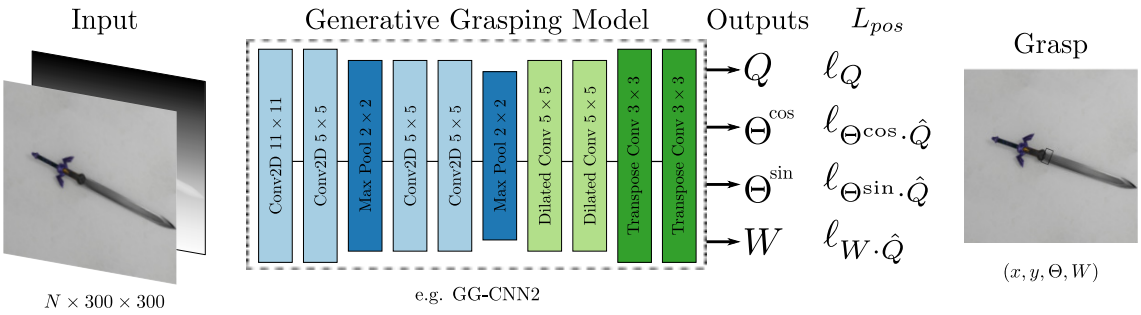


Figure 4.1: The proposed inference pipeline for generating a grasp proposal region using a generative grasping neural network. The angle and width grasp maps are scaled using a dot product between the output and ground truth grasp quality map. A grasp rectangle can be extracted in the same way as before by centring a grasp at the pixel with the highest predicted chance of success.

times to achieve higher accuracy on larger datasets. In order to grasp objects correctly, there should be a greater importance placed towards learning where successful grasps are likely to take place, compared to the correct gripper angle or width, as there is a much greater range of correct values if the grasp placement is correct. Since generative models typically only execute the highest quality grasp, the network should therefore be encouraged to learn better grasp placement.

As a result, in this chapter, a novel loss function is introduced which can be applied to the aforementioned generative grasping models that encourages better grasp placement. As outlined in Fig. 4.1, the *positional loss* is designed to focus model training on points where only suitable grasps occur to encourage efficient learning. Each loss ℓ is scaled by the grasp quality ground truth $\hat{Q}$, so the model learns about good grasp placement, whilst only learning about good grasp placement around these points. The network is therefore not penalised for errors on the angle or width associated with grasps at positions where the grasp is unsuccessful. By replacing traditional loss functions with the more targeted *positional loss* function, we focus the learning of the network towards successful grasps.

Here, the advantages of applying this novel *positional loss* function are demonstrated in some common generative robotic grasping CNN models, including the GG-CNN [5,158] and MTG-CNN [178] network architectures, as well as the more recent GR-ConvNet [179] model, on the large Jacquard grasping dataset [129]. The main contribution of this chapter is therefore the introduction of a novel loss function that is labelled, which when applied to a generative network displays he following improvements:

- *accuracy* - generation of more accurate grasp proposals leading to higher performance across all types of input on the Jacquard grasping dataset (see Table. 4.1 and 4.2);

- *efficiency* - faster convergence of network parameters leading to improved learning of appropriate grasps by focusing attention on objects in the scene, rather than the background (see Fig. 4.8);
- *efficacy* - the positional loss function greatly increases performance when trained with a limited number of examples (see Table. 4.3).

4.1 Methodology

In this section, the generative models used in the previous chapter are briefly revisited, which formalise the intended differences, and then outlining the experiments used to evaluate the differences resulting from this new loss.

4.1.1 Generative Grasping Networks

Given the grasping problem as outlined in Section. 3.0.1, each generative models described outputs grasp proposals in the form of grasp maps G. Therefore, each model outputs according to Eq. 3.5, with a grasp quality score Q, the two unit vectors (sin & cos) of the gripper angle Θ, and gripper width W. In each instance, each generative model architecture described takes an input image I of $N \times 300 \times 300$ pixels where N is the number of input channels, and each grasp map output is the same size as the input image. This is then used to reconstruct grasps during post-processing.

The model architectures used in the following experiments, the GG-CNN2 [158] and MTG-CNN models (see Section 3.1.2 for detailed descriptions) are used, as well as the larger generative residual convolutional network (GR-ConvNet) [179] described below. Each network used utilises the loss function as used in the original experiments and is then compared with the same model as trained with the newly introduced *positional loss* function. In the case of GG-CNN2, and MTG-CNN models, the positional loss is therefore compared against the typical mean squared error (MSE) loss.

Generative Residual Convolutional Network

The generative residual convolutional network (GR-ConvNet) from [179] is briefly described alongside a number of variations to the model introduced in the same work. This model was originally trained using 4-channel RGB-D images, but can also be trained using RGB, and depth images. Therefore, each model is trained with each input modality to

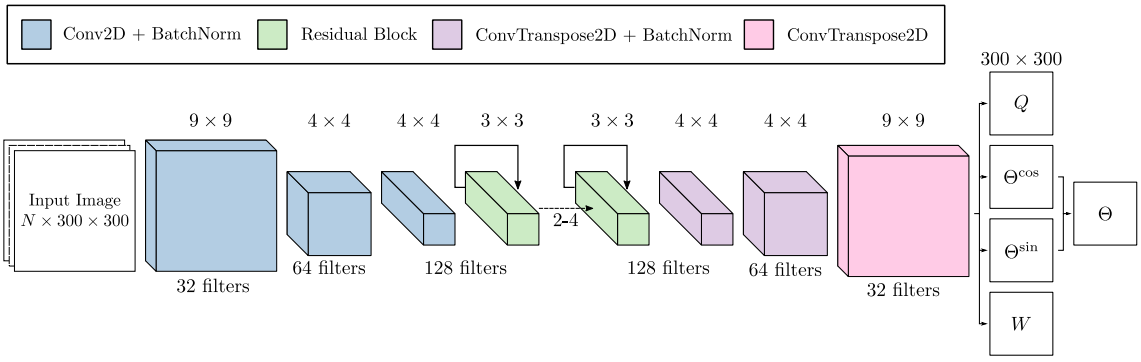


Figure 4.2: Generative Residual Convolutional Network (GR-ConvNet) architecture as featured in [179].

show that the new positional loss function improves performance no matter which type of input is used. Each variation uses the same general structure as in 4.2 with slight modifications:

- **GR-ConvNet** - The original GR-ConvNet variation, takes an input image which is processed by three 2D convolutional blocks with an increasing number of filters, from 32, to 64, and finally 128, before being fed into 5 residual layers. Each layer also containing 128 filters. This reduces the size of the image from 300×300 pixels to 75×75 pixels and the residual layers prevent vanishing gradients which diminishes accuracy. This is followed by a further three transpose convolutional layers with the reverse number of filters as the first three convolutional blocks, so that the four output images are presented at the same size as the input for clarity. All convolutional layers are followed by batch normalisation and rectified linear unit (ReLU) activation functions.

- **GR-ConvNet2** - as per above, except with an additional dropout [180] element which was set to a probability of 0.1 during training in the residual layers of the network as well as the output layers to reduce the chance of overfitting.

- **GR-ConvNet3** - as per above, except the dropout element is only present in the final output layers.

- **GR-ConvNet4** - the final iteration instead features an inverted structure with the number of filters at each layer getting smaller instead of larger. The structure is the same as the GR-ConvNet but the number of filters are inverted, decreasing from 32 to 16 and finally 8.

This produces a network with a total of 1,900,900 parameters, which is an order of magnitude greater than the other architectures, but is still smaller than comparative discriminative network architectures [43, 181, 182].

4.1.2 Positional Loss

Each network architecture is trained using the loss function used originally for each network and compared with the new positional loss function. Therefore, the GG-CNN2 and MTG-CNN networks are trained using mean squared error (i.e. MSE or L2) loss [37] and each GR-ConvNet is trained using Huber (i.e. smooth L1) loss [37, 183].

As stated in Section 4.1.1, each model outputs four per-pixel values which are trained using the same generated ground truth maps as in Section 3.1.2. This allows each model to be trained with a regression based loss by comparing the predicted model outputs $\mathcal{P}$ with the generated ground truth grasp maps $\mathcal{G}$.

In the case of the GG-CNN2 and MTG-CNN models that require MSE loss, the total computed loss from each output is represented by the following equation:

$$\ell_{MSE}(\mathcal{P}, \hat{\mathcal{G}}) = \frac{1}{N} \sum_{p \in \mathcal{P}, g \in \mathcal{G}}^{N} (p - \hat{g})^2, \tag{4.1}$$

where the total loss is the sum of all the pixels in the single predicted grasp p compared to the pixels in a given ground truth grasp $g \in \mathcal{G}$. From here, a hat operator $\hat{\mathcal{G}}$ represents the ground truth from the dataset for that target.

Therefore, the total loss of the grasp L_{grasp} used with the GG-CNN2 and MTG-CNN networks is the summation of each of the MSE loss components of the four parameter space output images $Q, \Theta^{\sin}, \Theta^{\cos}, W$:

$$L_{grasp} = \ell_{MSE}(Q, \hat{Q}) + \ell_{MSE}(\Theta^{\sin}, \hat{\Theta}^{\sin}) + \ell_{MSE}(\Theta^{\cos}, \hat{\Theta}^{\cos}) + \ell_{MSE}(W, \hat{W}). \tag{4.2}$$

with the total loss of the MTG-CNN network adding the total loss for the auxiliary task A, the same as in Section 3.2.2.

The GR-ConvNet on the other hand, preferred the smooth L1 loss, citing it to be the most stable and reducing the chance of overfitting. Therefore, the base loss function for

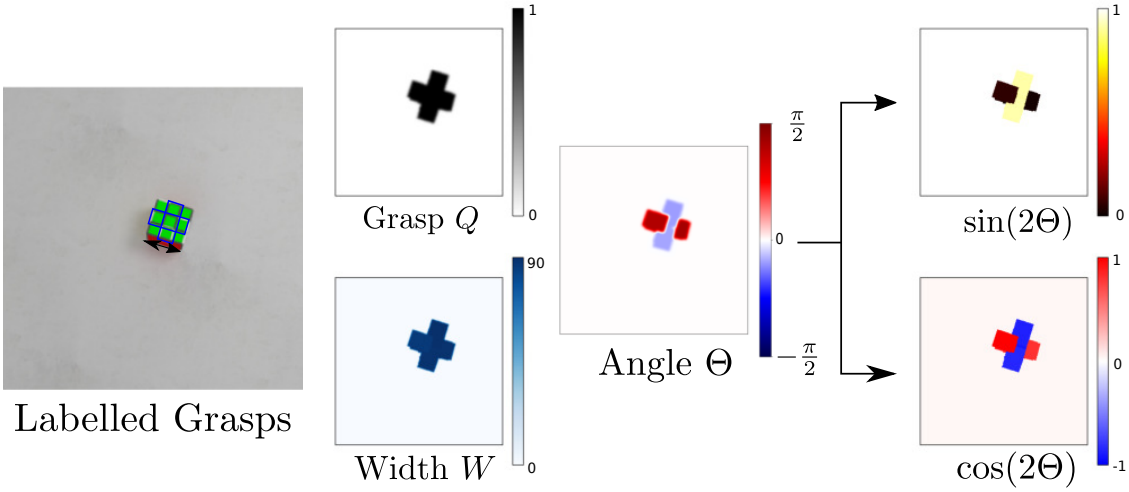


Figure 4.3: Component ground truth grasp maps $\hat{\mathcal{G}}$ generated from the Jacquard grasping dataset: grasp quality $\hat{Q}$, angle $\hat{\Theta}$, and width $\hat{W}$. $\hat{\Theta}$ is split into $\hat{\sin}$ and $\hat{\cos}$ for easier training.

this network according to [179] is represented as:

$$\ell_{L1}(\mathcal{P}, \hat{\mathcal{G}}) = \frac{1}{n} \sum^{k} z_k \tag{4.3}$$

where z_k is given by:

$$z_k = \begin{cases} 0.5(\mathcal{P} - \hat{\mathcal{G}}), & if|\mathcal{P} - \hat{\mathcal{G}}| < 1 \\ |\mathcal{P} - \hat{\mathcal{G}}| - 0.5 & otherwise \end{cases} \tag{4.4}$$

As shown in Fig. 4.3, in the Cornell dataset [78] and Jacquard datasets [129], the generated ground truth maps for $\hat{Q}$ are set to 1 at positions that resulted in a successful grasp and 0 elsewhere. Significantly, at positions where $\hat{Q} = 0$, the ground truth angle and width are also set to 0. As a result, the total loss L includes a contribution from the error in angle prediction and width prediction even at positions of the input image where no successful grasp is possible, for example, in regions containing the background not the object. The network must therefore learn to output values close to a default value of 0 for angle and width in such regions, even though grasps are unlikely to be attempted at these positions. This is because generative models typically centre grasps around pixel coordinates with the highest grasp quality score, i.e. $arg\,max(Q)$. Therefore, these angle and width parameters are not used. This places a significant burden on the network: typically the contribution of angle and width to the overall loss is equal to 75% with equal weighting, and early training epochs target reducing the predicted angle error at

background positions above other factors.

The positional loss function L_{pos} aims to scale the angle and width errors by ground truth for the quality of the grasp ($\hat{Q}$) at each position in the scene according to the following equation:

$$L_{pos} = \ell_{MSE}(Q, \hat{Q}) + \ell_{MSE}(\hat{Q}(\Theta^{\cos}, \hat{\Theta}^{\cos}) + \ell_{MSE}(\hat{Q}(\Theta^{\sin}, \hat{\Theta}^{\sin})) + \ell_{MSE}(\hat{Q}(W, \hat{W})) \quad (4.5)$$

By scaling of the loss in this manner focuses the learning of the network on grasp quality and encourages the network to only learn angles and widths only at positions where grasp attempts may occur making the contributions of these parts of the network much smaller, with a reduction of about 45%. The largest proportion of the total loss therefore becomes dedicated to learning good grasp locations and does not spend most resources learning to ignore data that does not contribute to the final grasp, as typically only a few best grasps are considered.

4.1.3 Experimental Setup

In order to compare between models trained with their original loss functions and the new *positional loss* function, each model is trained using the same methodology as in Section 3.2.2. A 5-fold cross validation is performed using each model on the Jacquard grasping dataset [129]. Each time 80% of the data is used for training, 10% of the data is reserved for a validation set, and the same final remaining 10% is reserved for testing. The ground truth examples are generated in the same way as [5] and Section 3.1.1(see Fig. 4.3).

Each model is trained using the original loss function intended for training according to the previous section(i.e. MSE loss for training GG-CNN2 [158] and the grasping branch of the MTG-CNN as in Section 3.1.2, and smooth L1 loss for the GR-ConvNet [179] variations). In their original works, each was trained with a different type of input. The GG-CCN2 network uses depth input, whereas the the GR-ConvNet utilises RGB-D input. However, both models can accept an N-channel image. Therefore, each model is trained using RGB, D, and RGB-D input to show that the positional loss improves performance regardless of the type of input used. Furthermore, experiments utilising the MTG-CNN architectures are trained to perform both saliency object detection and depth reconstruction as auxiliary tasks. The saliency object detection branch is also trained with

both MSE and binary cross entropy loss (BCEL).

As further evidence that the positional loss function helps to focus learning and generalise beyond the training examples, a few of the networks are also trained on the smaller Cornell grasping dataset [78], which features comparatively sparse training examples, and then tested on the same Jacquard test set. This is to show that with even a small number of training examples, the positional loss function improves accuracy whilst decreasing training time.

All the following models are also trained with the ADAM optimiser [176] on a NVidia GTX 1080Ti GPU.

4.2 Evaluation

Results are reported using a 5-fold cross validation method similar to that of Redmon *et al.* [77]. Performance is measured according to the rectangle (IoU) metric as detailed in Section 3.2.2, and the percentage of successful grasps on the Jacquard test set is compared between each generative grasping model trained with the loss function as described in their original work, against the new *positional loss* function.

Firstly, models and data from the previous chapter are presented to highlight the benefit of applying this simple change to training. Then, Jacquard IoU test results are presented for the GR-ConvNet models to show the generalisability of this new loss to larger architectures. Finally, the best performing models are trained on the entirety of the smaller Cornell dataset and tested on the same Jacquard test set to show the advantages of generalising from only a small number of training samples.

The qualitative differences of training various generative models are then inspected to analyse the impact of the alternative loss function, which go towards explaining improvements to performance.

4.2.1 GG-CNN2 and MTG-CNN Models

For the first set of experiments, the results of the GG-CNN2 and MTG-CNN models from the previous chapter are re-presented alongside the results from training these models with the new *positional loss* function in order to facilitate a convenient side-by-side comparison. The average number of correct grasps and standard deviation of these across the five test folds is shown in Table 4.1. From these results, training with the positional loss function

Table 4.1: Average predicted correct grasps for the GG-CNN2 and MTG-CNN models evaluated on the Jacquard grasping dataset with each loss function. Arrows indicate direction of performance increase.

Model	Input	Gr.Loss	Aux.Loss	MT.Loss	Grasps % (↑)	MAE (↓)
GG-CNN2 [47]	RGB	MSE	-	-	72.04 ± 3.44	-
	Depth				82.26 ± 3.76	-
	RGB-D				83.11 ± 1.88	-
	RGB	Pos	-	-	78.92 ± 0.97	-
	Depth				83.34 ± 2.41	-
	RGB-D				$\mathbf{84.11 \pm 1.44}$	-
MTG-CNN Saliency	RGB	MSE	MSE	×	78.61 ± 0.52	0.011
		Pos			$\mathbf{79.49 \pm 1.54}$	0.010
		MSE	BCEL		78.48 ± 0.97	**0.007**
		Pos			79.17 ± 0.77	**0.007**
	RGB	MSE	MSE	✓	76.77 ± 1.37	0.009
		Pos			79.34 ± 1.42	0.015
		MSE	BCEL		78.12 ± 1.00	0.008
		Pos			76.63 ± 1.28	0.008
MTG-CNN Depth	RGB	MSE	MSE	×	78.14 ± 0.65	0.022
		Pos			79.12 ± 1.40	**0.014**
	RGB	MSE	MSE	✓	79.53 ± 1.43	0.020
		Pos			$\mathbf{80.19 \pm 1.14}$	0.015

results in nearly universal improvement across all models. The best grasping performance for each model is marked in bold typeface, and is represented solely by models trained with the new loss function. Therefore, models trained with this loss identify a greater number of correct grasps compared to training with the previous baseline MSE loss.

For example, the highest performance on the Jacquard test set is reported by the GG-CNN2 model trained with the most information rich RGB-D input. However, the positional loss function further increases performance from 83.11% to 84.11%. Furthermore, this effect applies to both depth and RGB input exclusively, where grasping performance increases from an average of 72.04% to 78.92% for RGB input and 82.26% to 83.34% for depth input. This also reduces the variance in grasp performance, making training more stable, as shown by the standard deviation of grasping performance. Random examples of grasps on the Jacquard test set, produced by the GG-CNN2 model trained with positional loss, are shown in Fig. 4.4.

Similarly, the MTG-CNN model also produces more accurate grasp proposals when trained with the positional loss function regardless of the auxiliary function used. When

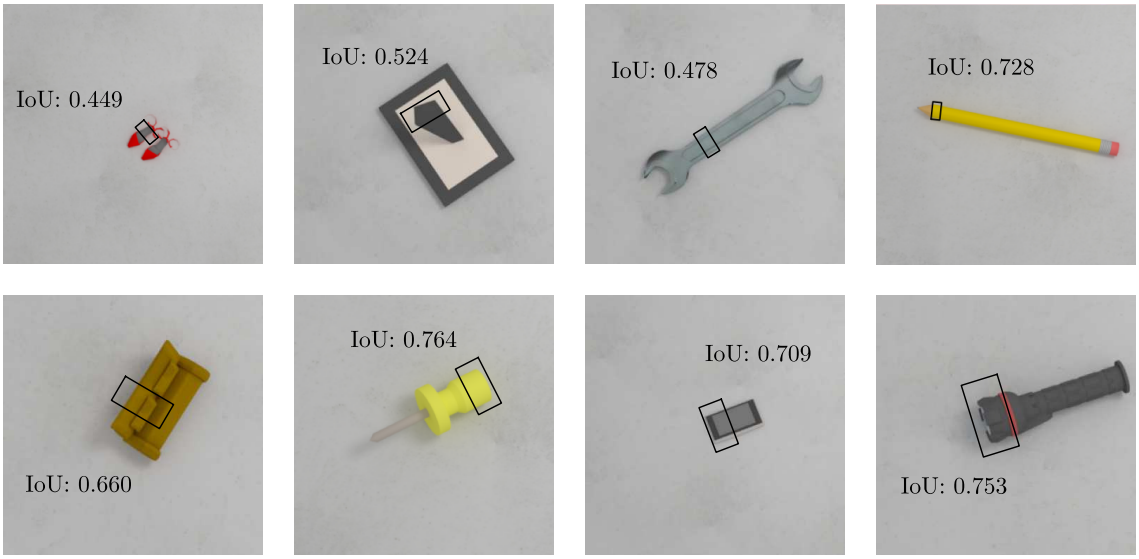


Figure 4.4: Random predicted example grasps from the GG-CNN2 model trained with the positional loss function.

trained to perform auxiliary salient object detection, the most best performing MTG-CNN achieves 79.49% grasping success which is higher than the same model trained with the base MSE loss. This success rate is also much higher than the 72.04% grasp success rate with the GG-CNN2 model when trained with the same RGB input with MSE and 78.92% when trained with the positional loss. Therefore, the combination of both multi-task learning and the positional loss function is complementary to one another, resulting in higher performance than either technique individually.

Even with the addition of the new positional loss function, the MTG-CNN model still achieves higher grasp success rates when trained to perform depth reconstruction over SOD. This finding is consistent with the result of the previous chapter and auxiliary depth reconstruction achieves an 80.19% grasp success rate trained with RGB input. In this case, the addition of the positional loss function also results in slightly better depth reconstruction performance. This is likely due to the reduced contribution of the angle and width images to the overall loss. This effect is less prevalent in the saliency branch, likely because this task is easier to predict for the Jacquard test set, where all objects are presented on a blank background, and therefore performance is near perfect.

The reintroduction of the multi-task loss ($-\log \sigma^2$) however, results in mixed performance compared to the previous chapters results. These findings show that with the reintroduction of the multi-task loss, grasping performance decreases slightly with the SOD auxiliary branch but this effect is reversed when applied to depth reconstruction.

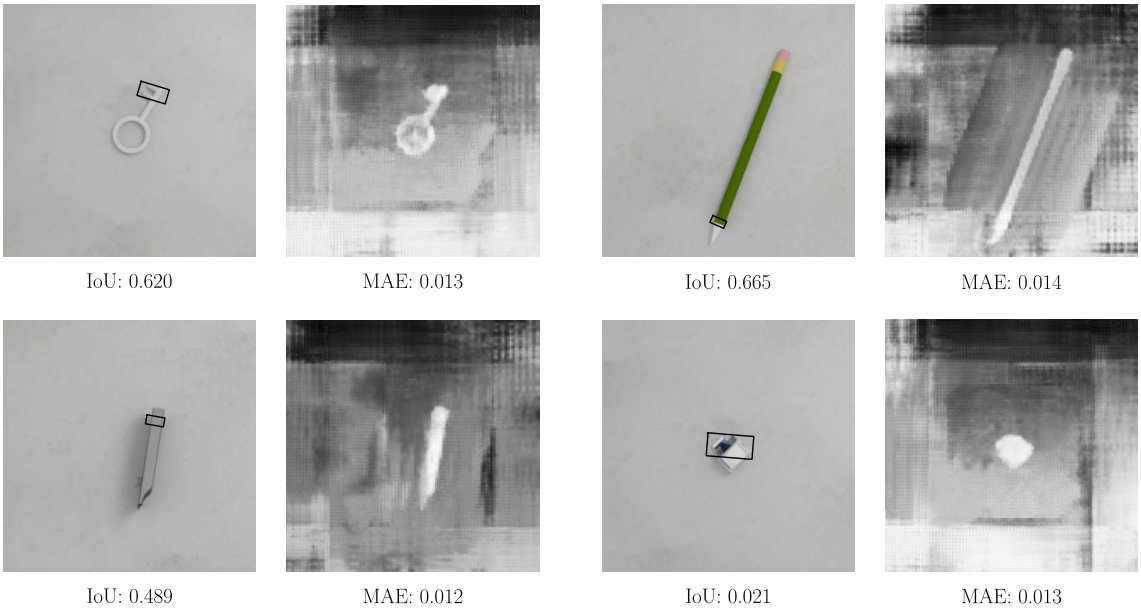


Figure 4.5: Random predicted grasp examples and depth outputs from the MTG-CNN model trained with the positional loss function.

As depth reconstruction likely provides more key information to improving grasp formation compared to SOD, there is an argument that optimising this relationship likely yields greater performance whereas optimising SOD past a certain point has a detrimental effect. Therefore, in future experiments, it is recommended to only apply this multi-task function when the auxiliary task is harder than the grasping task.

4.2.2 Generative Residual Convolutional Network

The following set of experiments detail the results of grasping performance with the GR-ConvNet and corresponding model variations from [179], as detailed in Section 4.1.1. Each model was trained using three different input modalities that are commonly used as inputs for robotic grasping tasks. This includes uni-modal inputs, such as RGB and Depth only images, as well as multi-modal RGB-D input. The results are displayed in Table. 4.2 where the best performing networks for each modality are marked in bold typeface.

From these results, we show that the positional loss function once again consistently achieves the highest performance across all input modalities, with the highest accuracy reported for the colour and depth only inputs represented by an average grasp success rate of 81.94% and 88.38% on the Jacquard test set using the GR-ConvNet2 model when trained with the new positional loss function. Additionally, this also achieved the highest grasp success rate with the RGB-D input, however this was for the base GR-ConvNet

Table 4.2: The average proportion of predicted correct grasps for the GR-ConvNet model [179] variations evaluated on the Jacquard grasping dataset with the new *positional loss* function.

Model	**Loss**	**Grasp Success (%)**		
		RGB	D	RGB-D
GR-ConvNet	Smooth L1	79.70 ± 0.79	87.08 ± 0.57	87.07 ± 0.47
	Positional	80.66 ± 1.40	87.19 ± 0.72	$\mathbf{87.46 \pm 1.07}$
GR-ConvNet2	Smooth L1	80.00 ± 0.98	84.49 ± 2.22	85.66 ± 1.74
	Positional	$\mathbf{81.94 \pm 1.17}$	$\mathbf{88.38 \pm 1.20}$	87.45 ± 1.17
GR-ConvNet3	Smooth L1	81.48 ± 1.44	87.18 ± 1.20	85.26 ± 2.96
	Positional	81.38 ± 1.58	86.03 ± 1.61	86.08 ± 0.70
GR-ConvNet4	Smooth L1	75.63 ± 1.30	83.30 ± 1.59	81.93 ± 3.49
	Positional	76.98 ± 2.99	83.30 ± 0.79	81.67 ± 1.48

with a success rate of 87.46%, only just outperforming the GR-ConvNet2 with 87.45%. A random set of grasps from the Jacquard test set using the best performing model, alongside the IoU with grasps from the ground-truth labels are presented in Fig. 4.6.

The new loss generally provides a performance increase across all the generative networks used in these experiments, in some cases providing an average 4% improvement over the base model using fully trained models, compared to the original smooth L1 loss function intended to be used with the network. The novel loss results in increases to overall performance to the GR-ConvNet networks, despite the larger number of parameters compared to the GG-CNN2 and MTG-CNN network architectures. Previously, the GR-ConvNet3 model achieved the highest grasping success rate with 81.48% and 87.18% using RGB and depth inputs respectively.

4.2.3 Limited Training Set

The final set of experiments aims to investigate the effect of training with the positional loss function when the number of training examples is limited. Therefore, a few of the best performing models are trained with their original intended input modalities from their respective work on the entire Cornell grasping dataset [78] (i.e. depth for the GG-CNN2 [47] and RGB-D for GR-ConvNet [179]). These results are shown in Table 4.3.

By training on a smaller dataset, and generalising to unseen data, the advantages of training using positional loss become clearer. Models trained with the positional loss function show higher accuracy, and by focusing the attention of the network, each model

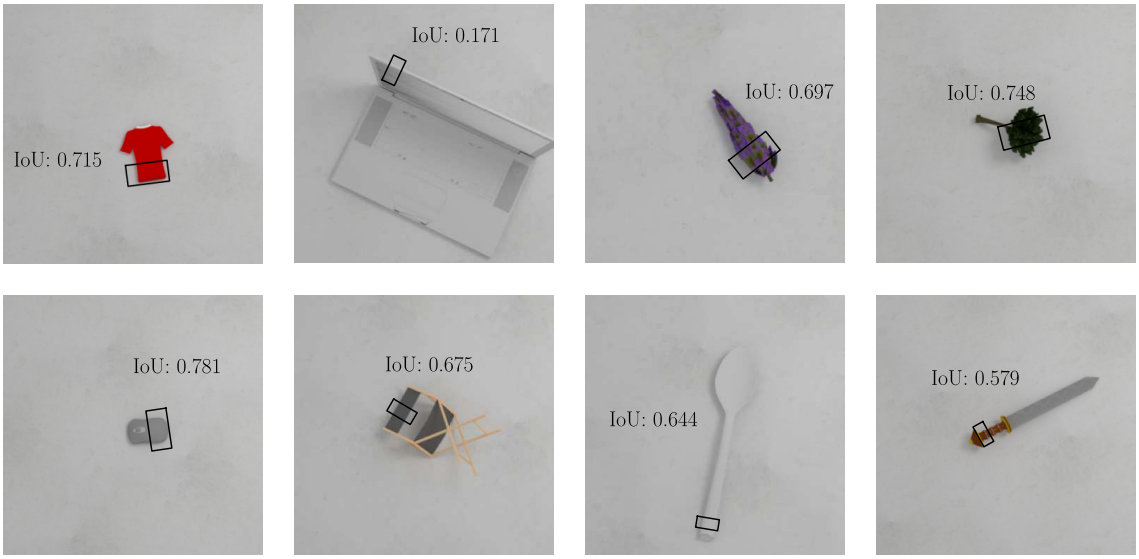


Figure 4.6: Random predicted grasp examples from the GR-ConvNet2 model trained with the positional loss function.

Table 4.3: Grasping success on the Jacquard test set for the GG-CNN2 and GR-ConvNet models when trained on the smaller Cornell Grasping Dataset.

Model	**Input**	**Loss**	**Successful Grasps (%)**
GG-CNN2 [47]	Depth	MSE	48.54
		Positional	**64.53**
GR-ConvNet [179]	RGB-D	Smooth L1	21.10
		Positional	37.01
GR-ConvNet2 [179]	RGB-D	Smooth L1	34.35
		Positional	49.46

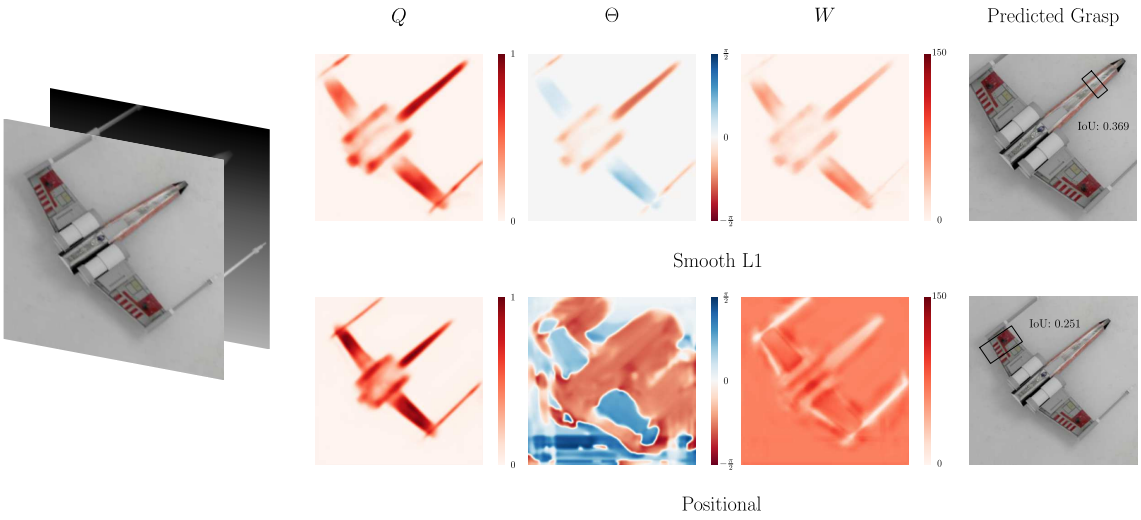


Figure 4.7: Comparison of example grasp map outputs from the GR-ConvNet2 model trained with smooth L1 versus positional loss.

is able to learn more from the smaller number of examples. This suggests that learning suitable locations for grasps is more important than learning the grasp angle and width. This is likely because there are a wide variety of suitable angles and widths which are valid for each suitable grasp centre pixel location, and so it is more beneficial and valuable to learn where the grasps are centred in the first place.

4.2.4 Effects of Loss

Alongside the improved grasping performance, evidence is presented to evaluate the quantitative and qualitative differences in grasping output, as well as a closer look at how the loss function alters training.

Effect on Grasp Maps

To investigate the reason behind the higher grasping performance success, we look directly at example grasp maps when trained with or without the new loss. An example of direct outputs from the same GR-ConvNet2 architecture is shown in 4.7.

As shown in this figure, due to the training method using the positional loss, there are a number of differences that can be qualitatively analysed. For example, whilst the quality score map loss ℓQ remains untouched during training, the average Q probability is higher when trained using positional loss. The resulting Q output is therefore generally more confident as to where grasps can be positioned, and considers more grasp centres as viable candidates. By reducing the contribution from the angle $\ell\Theta$ and width ℓW losses, the

model focuses on successful grasp placement. Therefore, the model is learning to predict grasps even in places where the original loss function does not.

As a result of only penalising loss values where grasps are likely to occur, the angle Θ and width W grasp maps also appear far more vivid than the original smooth L1 function. By ignoring irrelevant background information, the Θ and W maps therefore propose realistic values for every pixel in the scene, not just where grasps occur. This can be advantageous because generative grasp networks only execute grasps with the highest chance of success, i.e. $arg\,max(Q)$, therefore the network is not devoting resources to also learning to ignore background information and instead compartmentalising tasks more efficiently. Whereas the generic loss function learns to output no value on the background, the positional loss learns to propose an angle and width values for all pixel grasp centre locations. Therefore, it is likely that training with the positional loss function improves performance by considering grasp placements that traditional loss functions do not.

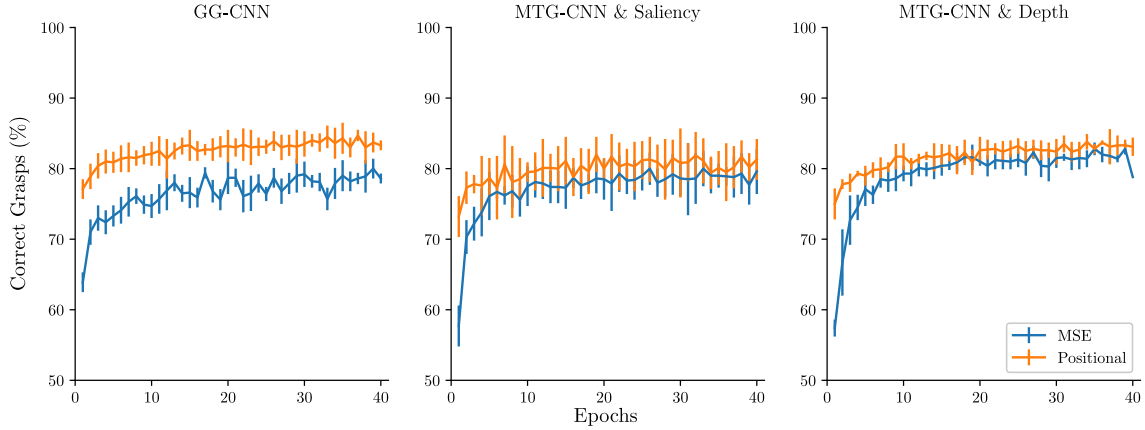
Training Stability

Another ancillary bonus from applying the positional loss function is that model training generally becomes more stable, especially in the first few epochs of training. By applying the new positional loss, attention is focused very early during training to only suitable grasp locations. This results in faster training and a higher average performance, even after only a few epochs of training, across all cross-validation trials.

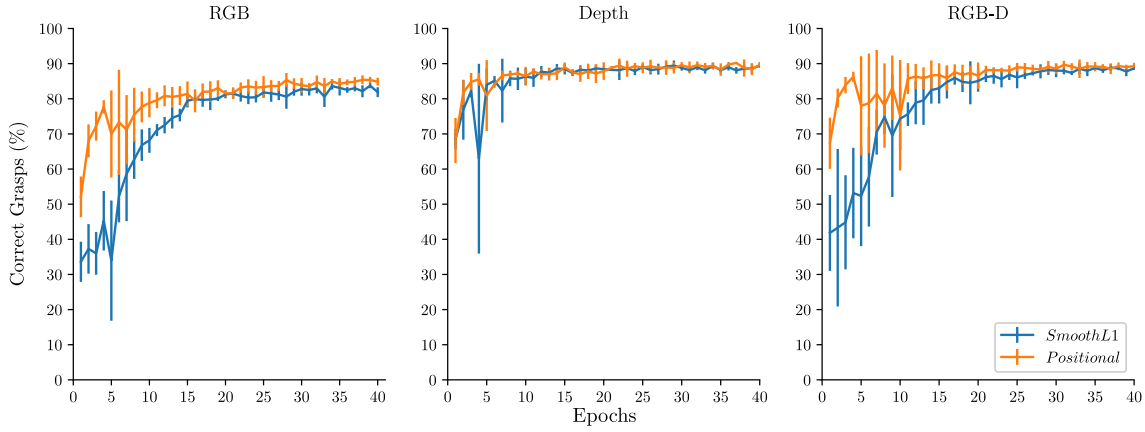
Fig. 4.8 presents grasping performance after each epoch during training on the Jacquard validation set. This shows the average model performance over time across all five cross-validation sets for some of the generative models presented. Similar to the grasping success on the test set, the positional loss consistently achieves the greatest grasping success rate reflecting general higher performance.

Initially, all networks trained on the new positional loss function perform better on average that the previous loss function. For example, the featured GG-CNN2 and MTG-CNN models trained with both auxiliary SOD and depth reconstruction tasks in Fig. 4.8a, quickly learning to distinguish what contributes towards a successful grasp. This effect is also replicated in Fig. 4.8c on the validation data using the GR-ConvNet2 model. This shows that the positional loss function provides a faster and better starting basis for learning grasps.

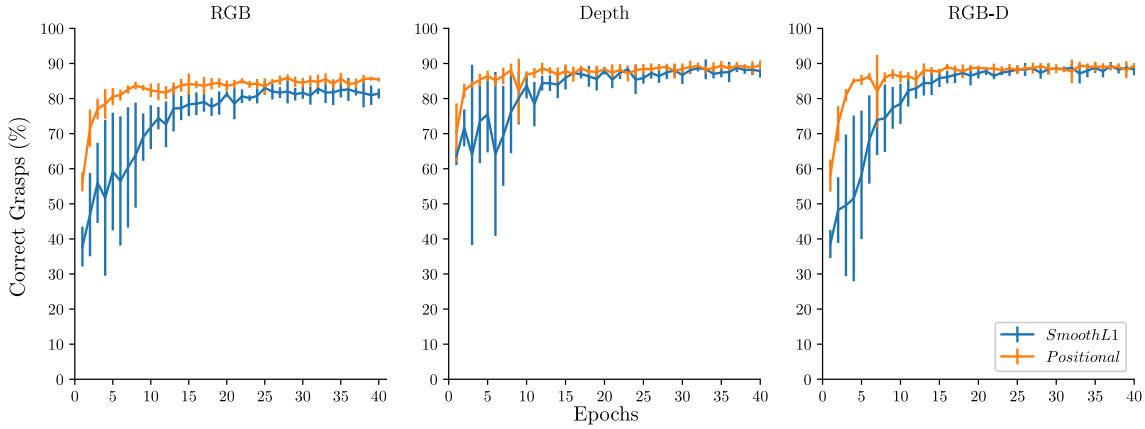
This further supports the hypothesis that it is more important to train the network to



(a) GG-CNN2 and MTG-CNN with RGB input



(b) GR-ConvNet



(c) GR-ConvNet2

Figure 4.8: Average grasping performance on the Jacquard validation set after each training epoch for different models trained with competing loss functions.

Table 4.4: Time, in milliseconds, to generate a grasp from an input image when trained using the positional loss function.

Model	Loss	Inference Time
GG-CNN2 [47]	MSE	$\sim$ 5 ms
	Positional	$\sim$ 5 ms
MTG-CNN [178]	MSE	$\sim$ 6 ms
	Positional	$\sim$ 6 ms
GR-ConvNet [179]	Smooth L1	$\sim$ 29 ms
	Positional	$\sim$ 29 ms
GR-ConvNet2 [179]	Smooth L1	$\sim$ 29 ms
	Positional	$\sim$ 28 ms

learn about where grasps are more likely to occur than the angle or width of the gripper. By focusing attention onto where grasps are more likely to occur, the loss encourages learning of suitable grasps early, providing a good starting basis for training generative models resulting in better performance overall.

Inference times

Another advantage of the introduction of this new loss for generative models results in generally increased grasping performance without impacting the overall inference time (see Table. 4.4). Since there is no increase in the number of network parameters or post-processing steps, there is no change to the amount of inference time taken compared to the base generative network to calculate the grasp proposal. This retains one of the best generative model advantages in that the inference time is still quick enough for open-loop grasp processing.

This results in much faster inference times compared to the alternative discriminative approach to grasping which means more grasps can be considered in the same amount of time. This is useful as it allows for rapid updating to changes in a scene. This means the model can cope with environmental problems such as the movement of objects in a scene. The reader is invited to refer to refer to Section 5.1.5 for examples of the successful deployment of the GG-CNN2 and GR-ConvNet models discussed in this chapter to a physical 6-DoF robot arm.

4.3 Summary

Despite generally high accuracy on common large grasping datasets, generative models that utilise grasp maps to propose appropriate gripper placements have relied on traditional loss functions for model training. This chapter shows that training networks with a bespoke and focused loss function designed for teaching grasping results in a higher grasping success rate due to the better placement of gripper positioning.

The intuition behind this loss was that learning the best placement for a grasp is more important than the given gripper angle or width, because there is a wide range of appropriate values for a given grasp centre. By using the common Jacquard grasping dataset [129], grasping success rate improved nearly universally across all network architectures and input modalities with this new loss, including when using the popular GG-CNN2 [158] model and MTG-CNN architecture from the previous chapter (See Table. 4.1), as well the larger GR-ConvNet [179] models (See Table.4.2).

Furthermore, this loss also resulted in faster and more stable training methods, as in Fig. 4.8, without impacting model inference times (See Table. 4.4), which allows for real-time control of a robot arm, as in other work like [5].

However, even though these generative models report high accuracy on these common datasets, they tend to only report findings according to the generally accepted IoU method. The next chapter investigates whether this metric alone is enough to predict high performance, and compares it against morre robust simulated and physical metrics.

CHAPTER 5

Gaussian Ground-Truth Grasp Maps

In previous chapters, a number of planar generative grasping models are discussed that are capable of proposing high-quality grasps for a wide variety of objects. The first network covered, the generative grasp convolutional neural network (GG-CNN) [5], was the first architecture to improve model performance and training time by learning to output separate components of a grasp in the form of *grasp maps*. This method generates a pixel-wise grasp quality representation which is reconstructed at test time into the more common form of representing grasping the *rectangle representation*, first defined in Jiang *et al.* [42] and further updated in follow up work by Lenz *et al.* [78] and Redmon *et al.* [77]: This represents a grasp as four parameters (x, y, Θ, W), the centre of a grasp in image pixel coordinates (x, y), the angle of the grasp Θ, and the width of the grasp W.

Up until now, the datasets used to train these models label successful grasps according to the rectangle metric described previously, and therefore a transformation is required to generate suitable ground truth grasp data used to train the grasp maps utilised by these generative networks. In previous chapters, a simple heuristic is implemented that generates grasp location data by assuming any grasp within the centre third (and approximately close angle) of an annotated successful grasp is valid. However, this assumption is incorrect: as Fig. 5.1 shows how grasps centred on pixels towards the edge of the centre third of the grasp rectangle can lead to gripper collisions when applied to a robotic arm, resulting in unsuccessful grasps on real world objects.

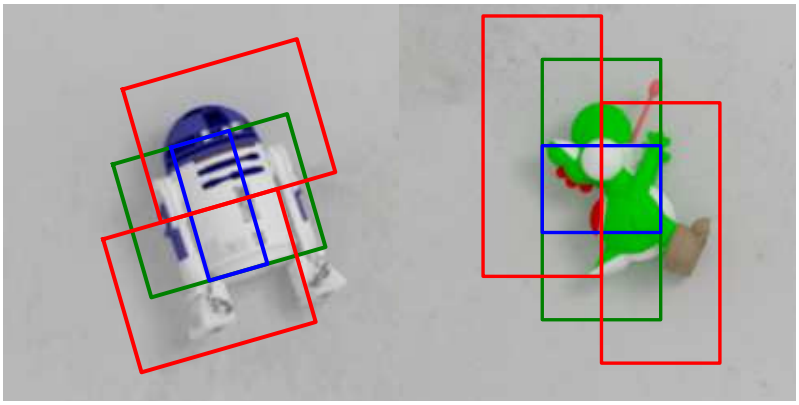


Figure 5.1: Current generative models for robotic grasping assume a binary representation of grasp labelling. Models are trained to recognise that any grasp centred on a pixel that falls within the centre third (blue) of a correct grasp rectangle (green) are suitable. However, grasps centred on the pixels closer to the edge of the rectangle are less reliable and result in collisions due to incorrect labelling (red).

To address this inconsistency, this chapter presents a modified Gaussian ground truth to train common generative grasping networks, which more closely resembles that of the training data based on recent work from, the orientation attentive grasp synthesis framework (ORANGE) [184]. By introducing this modified heuristic for training, the network learns to predict grasps that more closely resemble grasps previously successful in a simulated setting. This is more likely to generate more accurate and successful grasps plans which can better capture the geometric properties of the grasp pose to avoid potential collisions between the gripper and the grasped object.

We argue that reported performance according to the commonly accepted IoU metric, that implements an arbitrary limit of 25% overlap with any ground truth grasp rectangle, is insufficient for predicted real-world performance and suffers from the same problems that are present using the binary heuristic for training generative models. This results in inflated success rates on grasping datasets when measuring grasping performance offline because it does not penalise cases where grasps are unsuccessful due to factors such as a gripper collision.

This chapter therefore presents auxiliary metrics that are recommended to be presented alongside established grasping model performance to inform researchers in choosing the best performing models when transferred to real robotic arms. As such, model performance is also presented alongside widely accessible simulated and physical grasping benchmarks benchmarks to show these models are capable of better transferring to unseen objects. The contributions of this chapter are therefore as follows:

- *Gaussian training grasp maps* - a novel Gaussian ground truth is introduced which encourages generative models to produce grasps which more closely resemble the simulated successful grasps;
- *Simulated grasp benchmarks* - previous work has mostly reported grasp success on widely varying physical benchmarks. We demonstrate the increased accuracy for representing grasp success on an easily accessible and comparable simulated grasp trial (SGT) benchmark [129] for future studies to compare with;
- *Physical grasp benchmarks* - an easily reproducible simple robotic arm implementation is used to demonstrate that the model trained on simulated data is capable of direct deployment to a physical robotic arm without the need for any transfer learning. This model achieves a high grasp success rate on a previously unseen dataset of standardised 3D printed objects.

5.1 Methodology

To explain the rationale behind the changes to the generated ground truth grasp maps from the previous chapters, first we describe the changes to the generative grasp networks in line with current work. As a result, the changes to the generated ground truth from the dataset are defined and the updated training methods. Finally, we detail an example method of transferring the output of the network into a real-world 6-DoF robot, which is used to perform physical experiments with unseen objects.

5.1.1 Networks and Outputs

In the previous chapters, examples of generative grasping models were outlined including the GG-CNN2 [47] (see Section. 3.1.2), and GR-ConvNet [179] (see Section. 4.1.1). Each of these network architectures generate a grasp for each pixel in the scene in the form of *grasp maps* (previously covered in Sections 3.1.2 and 4.1.1. This creates an easy way to train models which contain a value for each pixel, with a grasp represented different by four outputs that can be reconstructed into a grasping rectangle during post-processing. These outputs are labelled: $Q, \Theta^{\cos}, \Theta^{\sin}$, and W as in Fig. 5.2. To extract a grasp proposal, the centre of the rectangle is assumed to be the pixel position providing the maximum Q value and use the corresponding angle Θ and width W from the same pixel position to generate a grasp.

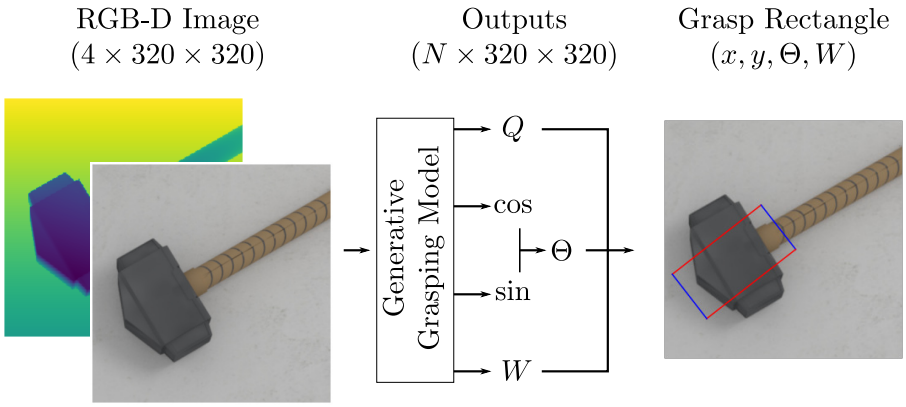


Figure 5.2: Given an RGB-D ($4 \times 320 \times 320$) image, each generative grasping model outputs four *grasp maps*: Q, cos, sin, and W of size $N \times 320 \times 320$ with N representing the number of output bins.

For previous model implementations, only a single ground truth map was generated for each object instance, corresponding to Q, $\Theta^{\cos}$, $\Theta^{\sin}$, and W grasp components. Due to this method, the angle and width correspond to a successful grasp centred at the pixel position in grasp map Q, and are arbitrarily set to 0 at positions that do not correspond to a successful grasp. However, because to the structure of the Jacquard dataset, a grasp map can contain multiple grasps centred on the same pixel that are all valid. When using a single grasp map, an arbitrary selection of which angle and width to use at such pixels must be made which drastically alters model performance [184].

In order to reduce overlapping labelled grasps where multiple grasps at different angles are centred on the same point, we employ a technique from the orientation attentive grasp synthesis model (ORANGE) [184]. This separates the grasp angles into N bins with each bin containing a range of $180/N$ degrees. The network then outputs grasp maps for each bin, which allows the network to learn N grasps at each pixel. The output of the network therefore becomes:

$$\mathbf{G} = (\mathbf{Q}, \Theta^{\cos}, \Theta^{\sin}, \mathbf{W})^{N \times h \times w}, \tag{5.1}$$

where each of the N dimensions gives the grasp maps restricted to that bin of angles. We compare the models that output grasps as a single bin and when split 1 into 3-bins.

In this instance, to reconstruct a grasping rectangle for testing, the maximum Q value across all three bins is taken as the (x, y) grasp centre, with the corresponding $\Theta^{\sin}$, $\Theta^{\cos}$, and W pixel values from the same bin output making up the final components of the grasping rectangle. For remaining overlaps, the grasp with the smallest width was used to generate the ground truth, in the same way to [184].

This work also showed that there are diminishing returns as the number of output bins increases. While there is an increase in performance from 1 to 3 bins, however 6 bins results in only a slight increase in performance over 3 except it vastly increases the training time and resources for relatively little improvement. Therefore, in the following work, we apply this technique to the generative grasping CNN (GG-CNN2) [158], and generative residual convolutional network (GR-ConvNet) [179] from previous chapters, as well the image detection model U-Net [13] also used by Chalvatzaki *et al.* [184]. All models are trained using 320×320 4-channel RGB-D images. Input data is cropped, resized, and normalised before being processed by the network to match the training data used and depth data is inpainted according to [5, 175]. The outputs are therefore also the same size as the input $N \times 320 \times 320$, where N is either 1 or 3.

5.1.2 Gaussian Ground Truth Grasp Maps

In the previous chapters, ground truth training data had to be transformed from the raw grasping data into a format suitable for training models that output grasp maps. In the Cornell [78], and Jacquard grasping datasets [129], data was collected using a robotic arm to gather suitable grasp points for each object. This included a physical arm in the case of the Cornell dataset, and a simulated arm in the case of the Jacquard dataset to gather more training examples. In each case, successful grasps are presented in the form of the common rectangle format as described in [42] and previously defined in Section. 3.0.1:

$$g = (x, y, \Theta, h, w), \tag{5.2}$$

each grasp centre is represented in pixel coordinates (x, y), alongside a gripper angle Θ, and corresponding gripper height h and width w using a parallel plate end-effector.

Of the generative grasping networks covered so far, the GGCNN2 [158] and GR-ConvNet [179], uses a heuristic that transforms the rectangle representation into a grasp map representation. In order to train suitable grasp locations, the traditional *binary* Q grasp map assumes that all pixels within the centre third of a grasp rectangle are correct grasps, and assigns a ground truth Q value of 1 if a pixel falls within this section of any grasping rectangle and 0 otherwise, as in Fig. 5.3. This heuristic is likely to result in inaccuracies such as grasps that are centred away from the object, and can result in gripper collisions (illustrated in Fig 5.1).

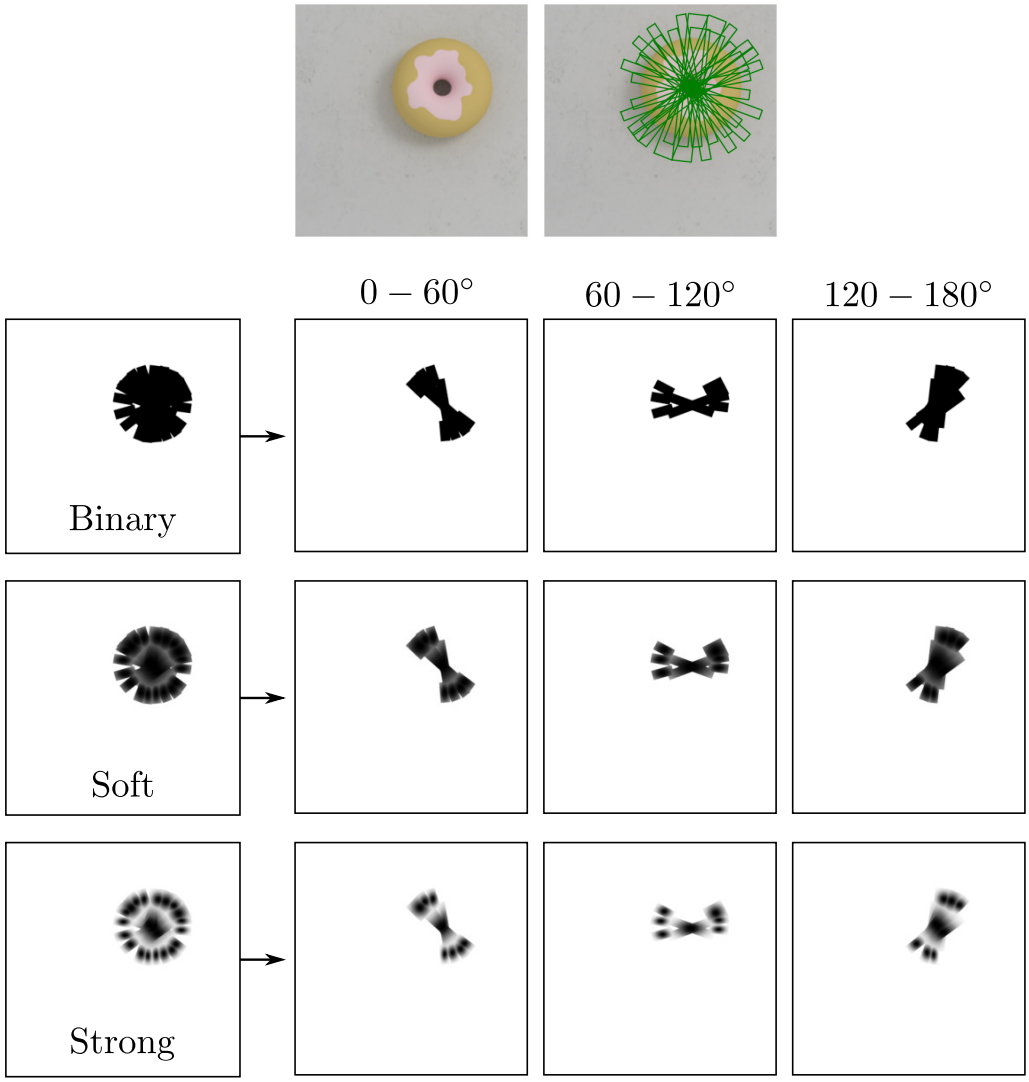


Figure 5.3: Comparison between example binary [5], soft quality [184], and strong quality ground truth grasp maps. A binary representation assumes any grasp within the centre third will likely be successful, which is given a value of 1, and 0 otherwise. Using the soft grasp quality score [184], the centre third of the grasping rectangle is still assumed to be successful, but values closer to the centre are higher than those at the edges, decaying towards a high threshold. Our strong quality proposal assumes edge values result in inconsistent grasps and removes the threshold, instead the ground truth decays towards zero at a rate defined by a hyperparameter σ. Each quality map is separated into 3 angle specific bins to predict the grasp quality score for the associated range of grasp angles using the method from Chalvatzaki *et al.* [184].

Chalvatzaki *et al.* [184] instead proposes a variation for generating ground truth maps from successful grasps. In this version, as only the centre pixel of a labelled grasp is certainly known to result in success, then only this value is assigned value 1. However, because neighbouring pixels are still highly likely to result in a grasp, then grasp quality scores gradually decays according to a Gaussian $\mathcal{N}$ distribution. The centre of the grasping rectangle has Q value 1, which fades away to a minimum value according to the equation:

$$Q(x,y) = \max_{g} \left\{ \min \left\{ \frac{\mathcal{N}(d,\, \sigma^2)}{\mathcal{N}(0,\, \sigma^2)} \delta, 0.9\delta \right\} \right\}, \tag{5.3}$$

where the Q ground truth maximum is generated over all annotated grasps g. $d = d((x,y),g)$ is the distance of the pixel (x,y) from the centre of the grasp g. $\delta = \delta((x,y),g)$ is an indicator function taking value 1 if (x,y) is in the centre third of the grasping rectangle of g and value 0 otherwise, and σ is the hyperparameter determining the strength of the Gaussian. In this case $\sigma = 2$ according to the ORANGE model and the minimum value is set to 0.9 within the centre third of the grasping rectangle. This method is referred to as the *soft* quality map, as in Fig. 5.3. This ensures that network is taught the centre of the grasping rectangle is a better location for grasp centre approximation.

However, this technique still considers all of the centre third of the grasping rectangle to be valid, which we argue is still likely to train the network to recognise grasps centred at unsuitable places around the object and more likely to result in gripper collisions. Therefore, we propose a alternative Gaussian heuristic for generating Q ground truth grasp maps, referred to as the *strong* quality map, defined in the following equation:

$$Q(x,y) = \max_{g} \left\{ \frac{\mathcal{N}(d,\, \sigma^2)}{\mathcal{N}(0,\, \sigma^2)} \times \delta \right\}. \tag{5.4}$$

This technique removes the minimum floor from the previous soft quality map, only including grasp centres as close to the original ground truth as possible.

We experiment with the strength of the Gaussian distribution $\mathcal{N}$, using the hyperparameter σ. This value alters the how sharply the $\mathcal{N}$ distribution decays the grasp quality $\hat{Q}$ value away from the centre. A smaller σ value represents a smaller standard deviation and closely focuses attention on the centres of successful simulated grasps, whereas a large σ allows the network to infer successful grasps further away from successful simulated locations.

Due to overlapping labelled grasps within a scene, the maximum value for a given

pixel across each angle bin is used. For multiple grasps using the same instances of a grasp centre and angle centred on the same pixel but using different jaw sizes, the smallest jaw size was chosen to match the boundaries of the objects shape more closely.

5.1.3 Simulated Grasp Trials

Until this point, model performance has been measured according to the commonly used Intersection over Union (IoU) method, also known as the rectangle metric. This is a fast offline method for assessing model performance as it can be evaluated locally according to the following criteria:

- the predicted grasp rectangle and a corresponding ground truth grasp rectangle share an IoU score of greater than 25%, and
- the offset of the predicted grasp rectangle aligns within 30° with the corresponding ground truth grasp rectangle.

Based on work from Jiang *et al.* [42], Lenz *et al.* [78] reduced the threshold for a grasp to be considered successful from 50% to 25%, arguing:

> "Since a ground truth rectangle can define a large space of graspable rectangles (e.g. covering the entire length of a pen), we consider a prediction to be correct if it scores at least 25% by this metric."

The threshold of 25% has been used to report performance in subsequent studies. However, as previously mentioned, this can lead to inaccuracies as a proposed grasp can meet the criteria for an absolute threshold dictating the rectangle metric, but would cause a gripper to collide with, or miss an object completely, during deployment [185]. The red rectangles shown in Fig. 5.1, represent grasps that fail to pick up the objects in simulation, but have IoU scores of over 25% so would be reported as correct in most studies.

Preferably, grasping metrics should always be provided alongside either simulated or physical grasp data to mitigate this issue, and show that the proposed grasp would be successful. Most studies however, provide widely varying benchmarks, typically comparing between different types of robotic arms and the physical dataset being tested on. Therefore, instead of introducing an entirely new benchmark, we also present data using an existing widely available metric, to allow for direct comparisons with future work. Alongside the labelled simulated data, included as part of the Jacquard dataset, we also utilise

the Jacquard on-line simulation server[1]. This was presented in the original work where the Jacquard dataset was introduced [129] referred to as the simulated grasp trial (SGT) metric.

The SGT measure of performance is a more robust metric, conducted on the Jacquard on-line simulation server that performs the proposed grasp in the same simulated environment as the data collection [129]. This is more costly in time and computation than the IoU metric, but is more reliable as it considers other data such as gripper collisions. Therefore, the best models according to the IoU metric are submitted to the Jacquard on-line simulation server in order to obtain more accurate benchmark for evaluating and comparing robotic grasping performance.

5.1.4 Training Method

For training and testing, the dataset is split 90/10% between training and test sets according to the same methods used by [179, 184], with no data augmentation applied during either stage. This leaves a total of 5449 grasping scenes from the dataset to form the test set. The same test set is used to evaluate both the traditional IoU metric and simulated grasp trial-based (SGT) criterion.

Input data was once again normalised similar to previous chapters. Colour pixel values are normalised to the range of $[0, 1]$ before subtraction the image mean to zero-centre the image data. Depth data is also normalised to the range of $[-1, 1]$ before a zero-centre via mean subtraction and clamping values within this range. All models are trained using the ADAM optimser [176] and early stopping is used once model performance has saturated.

Models are trained with their original loss function as well as the positional loss function from Section 4.1.2 and using the orientation bucketing system from Chalvatzaki *et al.* [184]. This is to show that each technique is capable of generalising to physical grasps successfully. Therefore, the grasp map outputs for the GG-CNN2 and U-Net models are trained using an MSE loss function and the GR-ConvNet model is trained using smooth L1 loss, against the positional loss function.

Following [184], the losses are also scaled by multiplying them with the number of discretised angle bins N and thus making the overall loss for the network equal to:

$$L_{grasp} = N \times \left(\ell_Q + \ell_{\Theta^{\cos}} + \ell_{\Theta^{\sin}} + \ell_W\right), \tag{5.5}$$

[1]. Available at: https://jacquard.liris.cnrs.fr/

with L_{grasp} representing the total loss for the given network and ℓ representing the individual MSE or smooth L1 loss for the given network. The new positional loss L_P function is then given by:

$$L_P = N \times \left(\ell_Q + \hat{Q}(\ell_{\Theta^{\cos}} + \ell_{\Theta^{\sin}} + \ell_W) \right), \tag{5.6}$$

In generating the ground truth, for situations where multiple grasps centred on the same pixel values existed with different corresponding angles and widths: the smallest sized grasp is used, as in [184] to train the model to pick the grasp closest to the object. Similarly, a half jaw size is adopted during testing as in [186].

5.1.5 Robotic Implementation

Finally, in addition to the IoU and SGT metrics presented, experiments utilising a low-budget robot arm are implemented to show that the model can easily transfer to a physical real-world setup. The setup takes an image from above using an Intel RealSense SR300 RGB-D camera, in the same orientation of that used in the Jacquard dataset, and generates the given grasp proposal from the model for the given object.

The robot arm used in this work has 6 Degrees of Freedom (6-DoF) WidowX arm from Interbotix Labs: a 1-DoF rotating base, three 1-DoF joints, a 1-DoF rotating wrist, and a 1-DoF parallel plate gripper with minimum 1cm and maximum 3cm width. The setup is shown in Fig. 5.4 and is the same low-cost arm as used in REPLAB [18]. Grasp plan motions are created using ROS inverse kinematics and planned with the *MoveIt* package.

Using Eq. 3.6, the 2D output from the model is transformed into the robots frame of reference by taking the maximum pixel coordinate (x, y) from the grasp quality score, and using the corresponding depth coordinate from an RGB-D camera to the depth point in 3D space z to form a 3D grasp location (x, y, z).

To demonstrate that these trained models are capable of transferring knowledge and generalise to completely unrelated objects, a standardised set of 3D printed objects is used for testing called the Evolved Grasping Analysis Dataset (EGAD)[2] [158]. This features a diverse range of objects of varying difficulty and complexity, including simple and antagonistic examples. The dataset ranges from A0 representing the easiest and simplest object to grasp, to G6 representing the most difficult and complex object. Increasing lettering corresponds to increased complexity and increased numbering represents more difficult

[2] Available at https://dougsm.github.io/egad/

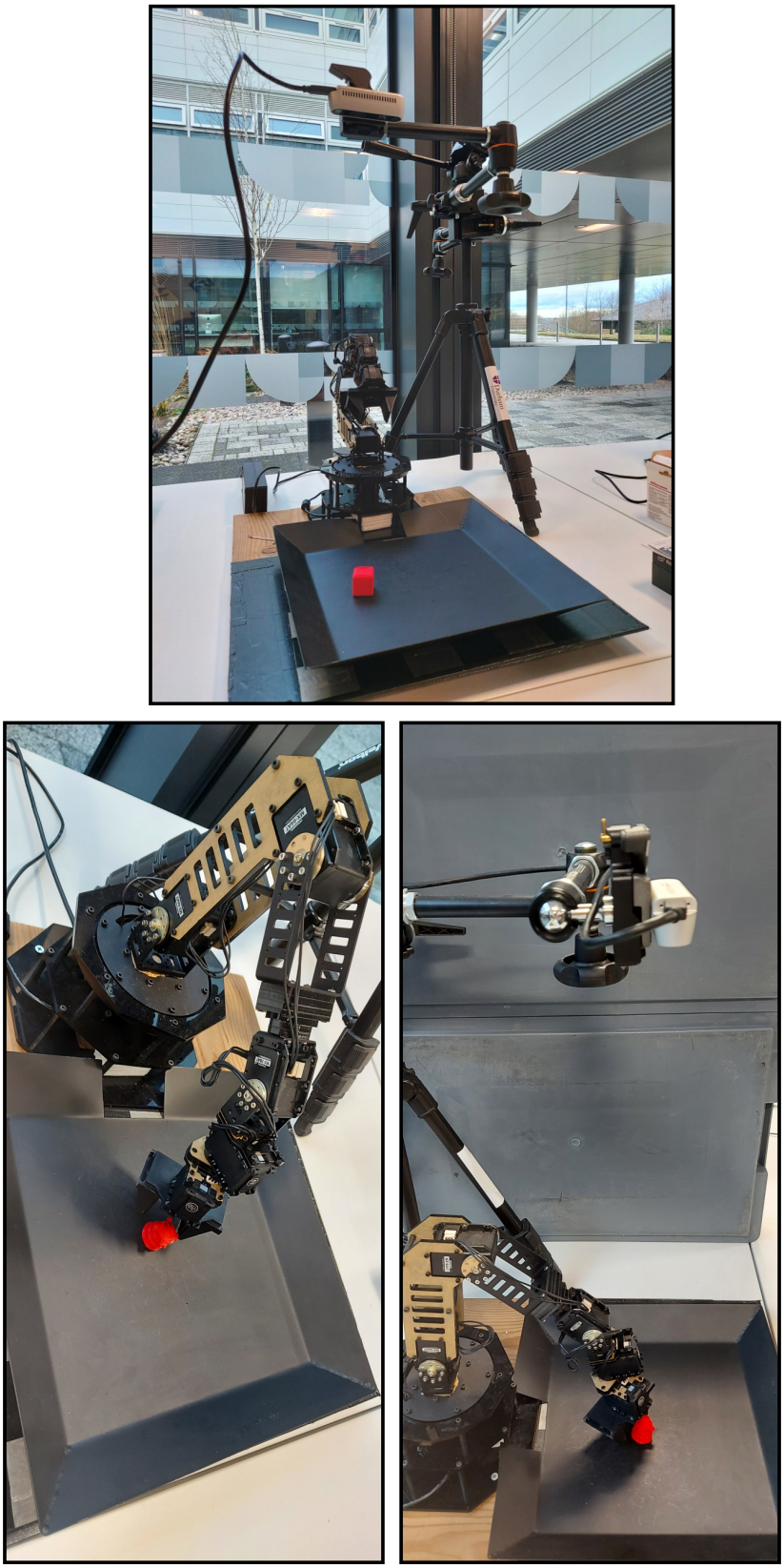


Figure 5.4: The setup of the WidowX robot arm used in the physical experiments, with the camera positioned above the scene.

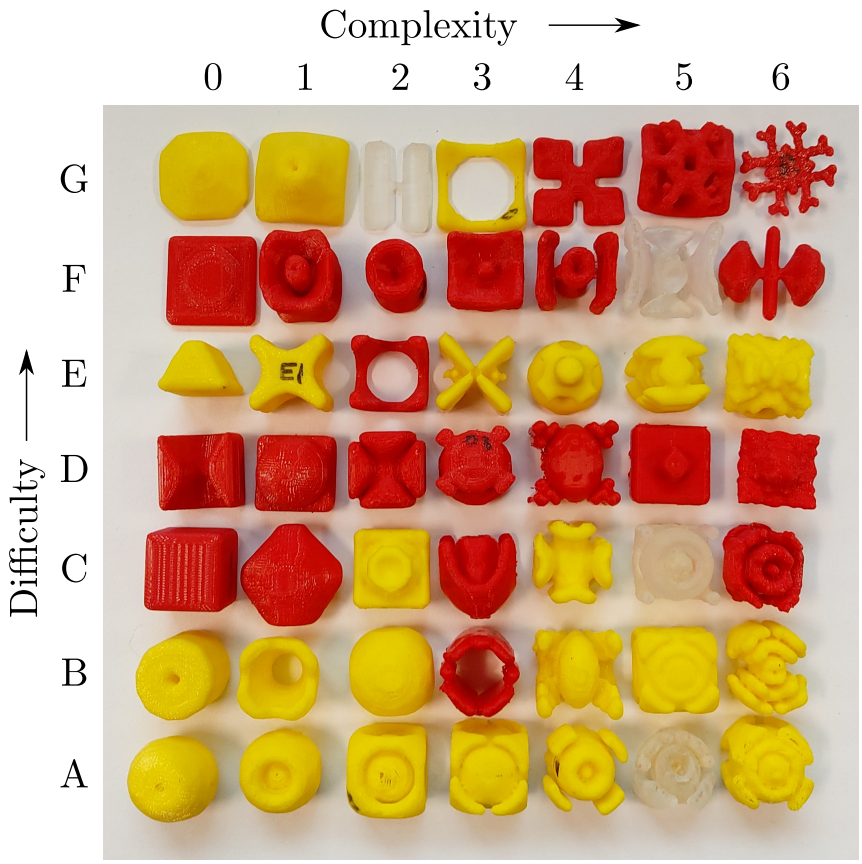


Figure 5.5: 3D printed EGAD object dataset [158]. Objects are strategically generated to represent a range in difficulty A-G from the easiest row A to the hardest difficulty G. Similarly, objects range in complexity by column from 0-6 with 0 being the simplest and 6 being the most complex to grasp. Objects are printed in a range of colours, or in resin where objects are more prone to breaking.

grasping objects.

The same grasping methodology as used in the original EGAD study [158] is repeated. Each object is thrown randomly into the arena and a grasp is attempted 20 times for each object. The grasp is considered successful if the object is lifted above the arena once the gripper has closed. The object is then dropped back randomly into the arena for the next attempt. If the object is unsuccessfully grasped then it is manually reset by throwing it back into the arena randomly, to ensure the network is not continuously attempting incorrect grasps.

Data is presented using the model as trained with the simulated Jacquard data with no transfer learning involved. This is to show ease of transferability and that the model can easily generalise to similar settings. For a description of the calibration process and transforming from the camera frame of reference to the robot frame of reference see Appendix B.

5.2 Evaluation

In this work, a series of generative grasping models including: the GG-CNN2 [158]; GR-ConvNet [179]; and U-Net architectures [13, 184], are trained on the Jacquard Grasping Dataset [129]. The results from both the IoU metric [42, 78] and SGT are reported for the same unseen data. The IoU metric is used as a quick offline metric for evaluating performance of all models and then the best performing models are tested using the simulated physics environment on the Jacquard test server [129], as this is a more robust evaluation of performance.

Each model is trained using the traditional *binary* ground truth grasp map introduced in [5], the *soft* quality map from [184], and the *strong* Gaussian grasp map, as described in subsection 5.1.2. The strength of the Gaussian filter σ is also varied to find the optimal spread of trainable parameters for the dataset.

5.2.1 Offline versus Simulated Performance

The performance of each model when measured using the typical IoU threshold of 25% is reported in Table 5.1. Firstly, this data shows that models trained with three output angle bins perform better than those limited to one angle bin, as previously shown in [184, 186]. Similarly, models trained with the *positional loss* function outperform the same model when trained with each respective base loss function, as previously shown in [178]. As far as we are aware, this is the first work to combine both these methods concurrently. This approach achieves the best reported IoU metric for each model, showing these two improvements complement one other, which has not been previously demonstrated. Since the evaluation of these models in simulation takes significantly longer to produce than the typical offline evaluation, only the highest performing models were evaluated, all models compared in the SGT results in Table 5.2 feature models trained with both methods in unison.

From the IoU measure of 25%, the conclusion would be that there is little difference when comparing the same models on different ground truth maps. The best reported value overall is achieved by the U-Net model with a generated binary ground truth (94.66%), which slightly outperforms the same model when trained with the strong (94.35%) Gaussian maps followed by the soft quality ground truth map (93.45%). This conclusion, however, does not hold when we consider the more robust SGT results in Table 5.2.

The results from this table show that when we analyse performance according to the

Table 5.1: Performance on the test portion of the Jacquard grasping dataset according to the IoU metric at the 25% threshold

Model	**Loss**	**Bins**	**Binary**	**Soft**	**Strong**			
			σ	2	2	1	0.5	0.25
GG-CNN2 [158]	MSE	1	**87.87**	87.69	86.79	87.50	86.86	85.74
		3	88.00	**88.73**	87.83	87.65	86.93	86.02
	Pos	1	91.21	**91.99**	88.13	90.18	90.18	89.96
		3	**93.98**	88.59	91.39	92.90	90.93	92.42
GR-ConvNet [179]	Smooth L1	1	**90.86**	90.82	90.16	90.77	89.74	89.10
		3	91.65	92.05	91.98	**92.40**	91.41	**92.40**
	Pos	1	92.27	91.89	91.76	91.47	91.98	**92.35**
		3	**93.69**	91.82	93.47	90.99	93.21	92.40
U-Net [13]	MSE	1	**90.55**	89.52	90.48	89.94	89.94	89.91
		3	**91.78**	91.14	90.51	89.21	89.67	89.89
	Pos	1	**93.61**	93.30	92.62	91.69	92.48	92.18
		3	**94.66**	93.45	94.04	94.35	93.83	92.59

Table 5.2: Predicted performance of models trained with three output bins and positional loss function on the Jacquard grasping dataset across a range of IoU thresholds and actual performance according the the SGT metric.

Model	$\hat{\mathbf{Q}}$	**IoU**						**SGT**
		25%	**30%**	**50%**	**75%**	**mIoU**	**AUC**	(%)
GG-CNN2 [5]	Binary	93.98	92.33	79.61	30.21	62.46	0.626	85.41
	Soft	88.59	85.15	69.41	25.49	57.40	0.575	85.43
	Strong	92.90	91.14	80.29	39.20	64.13	0.649	86.58
GR-ConvNet [179]	Binary	93.69	91.83	83.01	39.37	65.57	0.657	85.36
	Soft	91.15	88.81	79.52	23.47	61.16	0.615	83.06
	Strong	93.21	90.95	82.44	49.62	66.98	0.671	85.89
U-Net [13]	Binary	**94.66**	**93.25**	**84.42**	43.11	66.61	0.668	85.69
	Soft	93.45	91.43	82.27	40.03	65.05	0.652	85.78
	Strong	94.04	92.31	83.87	**50.71**	**67.69**	**0.675**	**87.94**

SGT score, the difference between the binary and Gaussian methods does not match predicted performance at the lower IoU thresholds. According to the SGT score, the strong Gaussian map now achieves the highest accuracy in each model on this more robust measure of performance. This is a slightly lower performance than the IoU metric but indicates which model will perform best in practice on a real robot arm, as it also considers factors more appropriate to a real setting, such as gripper collisions and grasps not included in the dataset. Here, the best performing U-Net model reports a total of 87.94% successful grasps on the same 5449 object scenes as used to measure the IoU compared to only 85.69% using the binary grasp map and 85.78 using the soft grasp map. This is over a 2% performance increase over the highest reported result for the SGT metric so far [185] .

Auxiliary information is also provided in this table to further investigate this difference in SGT performance. Despite reported close success rates at the traditional 25% threshold, the grasping performance of the models widely differs at higher thresholds. A higher performance at these larger IoU thresholds suggest the model is proposing grasps that highly resemble a grasp from that of the test set, whereas success at a lower threshold suggests that the proposed grasp matches any grasp and is liable to the incorrect grasps proposed in Fig. 5.1. The variation in performance between the models at the 75% threshold suggests that the binary and soft $\hat{Q}$ training maps, are not learning how to best recreate the training data. Therefore, the 25% threshold by itself is not enough to predict real world performance.

We therefore propose that when predicting grasp success in the future, instead of just the typical 25% (IoU@25) threshold, performance should also be reported according to the other metrics used in this table. This includes performance at a number of thresholds, as well as auxiliary information such as the mean IoU (mIoU) score, or the IoU area under curve (IoU-AUC) scores, which reveal similar information about the networks capability to learn from the data. Together, with the introduction of both these metrics including the range of thresholds and the mIoU/IoU-AUC scores as metrics, we aim to help researchers decipher which models perform better when transferred to a real setting, as the mIoU metric is better correlated with the final SGT performance (pearson's $r = 0.541, p = 0.132$), compared to just the IoU@25 metric (pearson's $r = 0.402, p = 0.283$).

Whilst the effects of altering the hyperparameter σ while using strong Gaussian maps are considered, there is no clear optimal value. Generally performance benefits from a moderate value in which the Gaussian map does not include the edges of the grasping

rectangle but maintains a large enough collection of high quality grasp centres to learn from. We note that the optimal Gaussian scaling is likely tied to the density of labelled grasps in the dataset for a given object and a given model. As the Jacquard dataset contains a high density of grasp labels, it is likely that the optimal scaling factor is smaller than for a dataset with more sparsely sampled grasp labels. As such, this hyperparameter may require fine tuning when training with other datasets.

To better illustrate the improvements made by these techniques, Fig. 5.6 also shows the interaction between the type of grasp ground truth map used for training, and the positional loss function across all IoU thresholds. Here, the U-Net model is used as an example across various training parameters.

In Fig 5.6-a, using only the initial training parameters, the model is able to report relatively high performance at the 25% threshold. However, by implementing the positional loss function, performance is improved at these lower thresholds, and which is reflected by shifting the grasp success rate upwards in the bottom half of graphs. This suggests that the positional loss is more often able to predict grasp proposals that line up with any labelled ground truth rectangles, as previously predicted, resulting in a higher predicted grasp success and grasps in more suitable locations.

On the other hand, training a model using our proposed strong Gaussian ground truth maps (Fig. 5.6-c,f,i,l), shifts the grasp success performance to the right in comparison to the binary (first column) and soft (second column) quality maps. This is especially apparent when models are trained in conjunction with the multiple output bins instead of the single one. Therefore, of the grasps proposed by these models, the grasp rectangles are more likely to highly intersect those of the ground truth, resulting in better grasp performance at higher thresholds because grasp proposals are better centred around labelled grasps. This is important as a higher intersection is less likely to result in success in gripper collisions which would fail in practice.

Subsequently in future work it is suggested that if a fast, offline estimate of performance is presented: the mean IoU (mIoU) score of proposed grasps should also be reported alongside the commonly used IoU metric, at the thresholds described above, as a predictor of robotic arm/SGT success. This average IoU metric provides insight to how the model performs regardless of a theoretical absolute threshold, when concurrent physical or simulated results are not provided for comparison with currently deployed models.

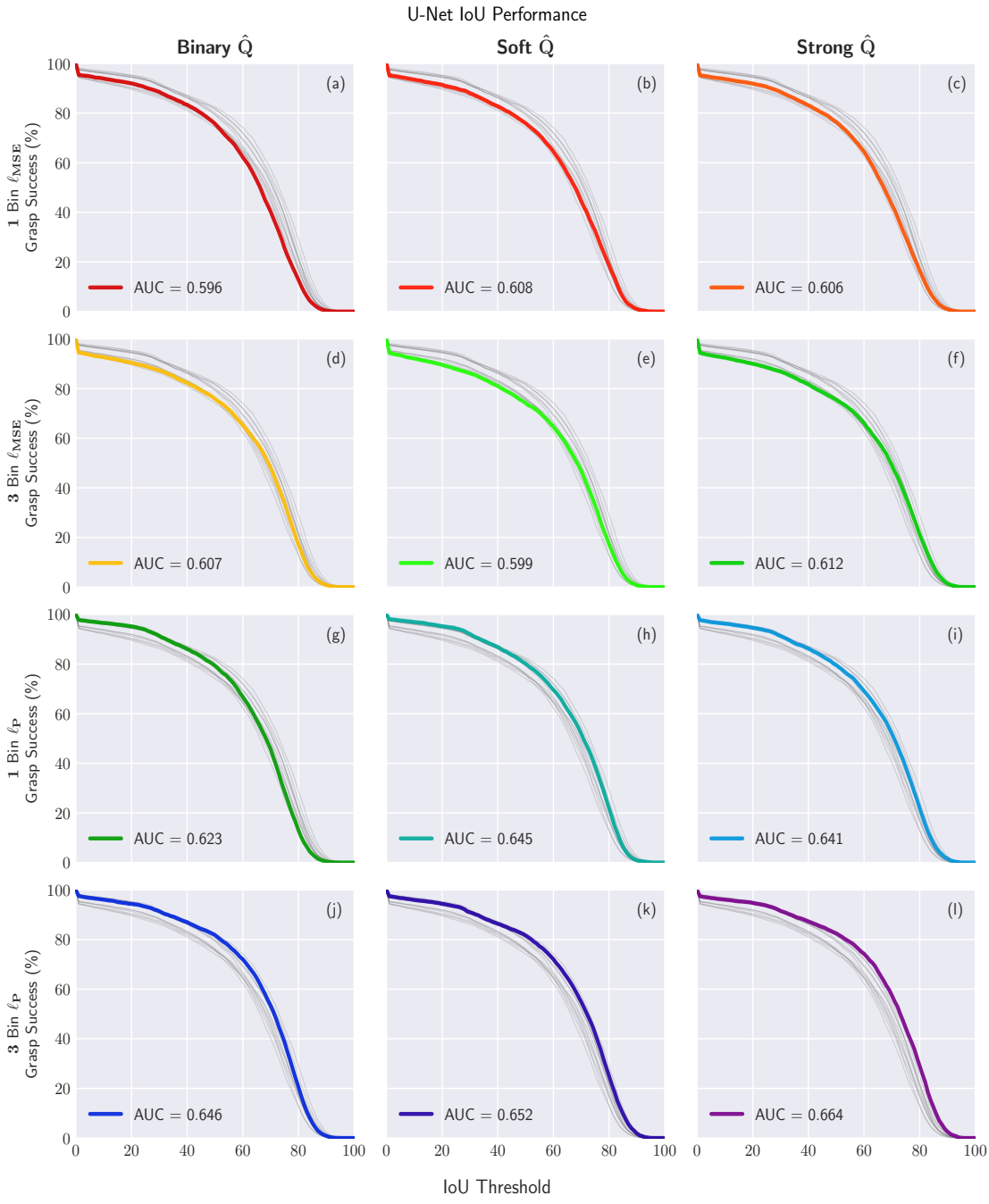


Figure 5.6: Reported grasp success as the IoU threshold is increased for the U-Net model trained with different ground truth maps $\hat{Q}$, altering the number of output bins, and when trained with the positional loss function. All models trained with the strong Gaussian $\hat{Q}$ ground truth are trained with $\sigma = 2$ for consistency. Each graph shows the success rate for the given parameters at different thresholds (in colour) overlaid on top of each other line (in grey) for easier comparison.

5.2.2 EGAD Results

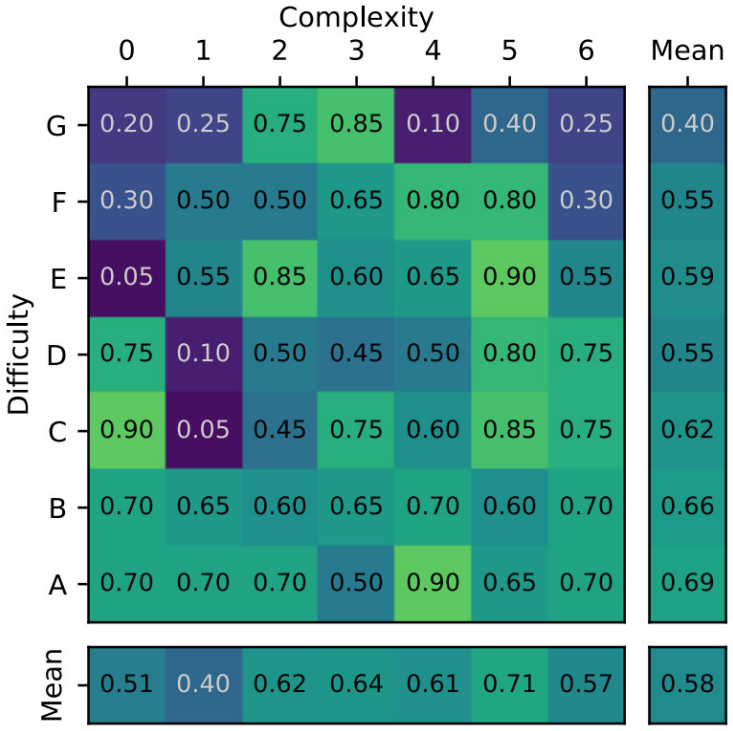
In addition to the SGTs, which show that the trained model is capable of producing grasps in the environment native to the training procedure, the model was also applied to a standard low-cost robotic arm. The arm is tasked with picking up each object from the EGAD [158] evaluation set 20 times for a total of 980 grasp trials.

A grasp is considered successful if a correct plan is made to attempt an object grasp and the arm is able to lift the object above the arena after closure of the gripper is made. We apply a single-shot method to grasping in which only one grasp plan is made in the same way as the simulated grasp trials take place. The camera is placed above the scene to mimic that of the Jacquard dataset but otherwise no transfer learning took place. The results of these tests, are shown in Fig. 5.7.

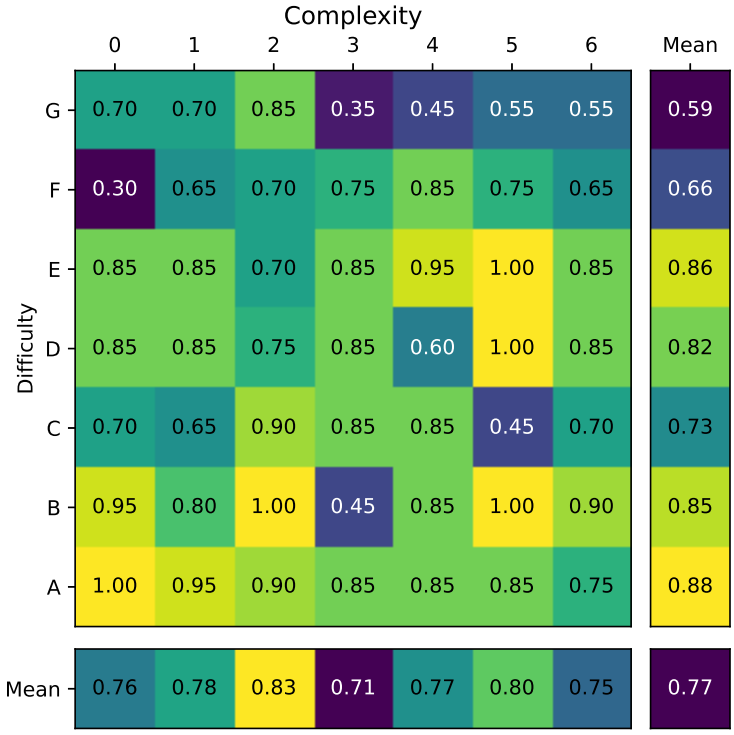
The applied model performs relatively well overall despite only training on simulated data. The model maintains reasonable consistency across all object complexities, and generally decreases in performance as object difficulty increases. The model performs best when grasping the easiest objects (A) and slightly dropping in performance towards the most difficult objects (G). In some trials, the robot is even able to achieve perfect or near perfect results, especially with easier objects. However, in all trials, the robot made an accurate attempt to grasp the object in the scene, which shows that the model is able to be applied with high accuracy without transfer learning. This results in an overall mean accuracy of 77% over all grasps attempted which, while not directly comparable due to the difference in arm setup, is higher than the 58% accuracy achieved by only the base GG-CNN model in the original study [158].

Most failure cases observed are due to the object difficulty due to grasping parts of the object that are purposefully difficult, such as angled sides or raised edges. See Fig. 5.8 for examples. Other cases where grasp success is low, such as object B3 or D4, resulted from a lack of knowledge about the gripper. The model would predict grasps that resulted in object collisions with the gripper plates by grasping along an unfriendly axis relative to the robot end-effector, cases which would otherwise be fine using a narrower pinch gripper. These could be improved by further training with the specific end-effector.

Fig. 5.9, Fig. 5.10, and Fig. 5.11 show some example grasps and model outputs when applying the 3 orientation bins. Without any transfer learning techniques, the model is able to generalise to these new objects and is able to distinguish between the multiple objects, a task previously untrained on. This shows this model can be directly applied with



(a) Results from the original EGAD study



(b) Results from our experiments.

Figure 5.7: Average grasp success rate for each object in the EGAD [158] evaluation dataset. Outer cells show the mean for that row and column. 5.7a shows the results from the original study compared to 5.7b the data collected in our experiments.

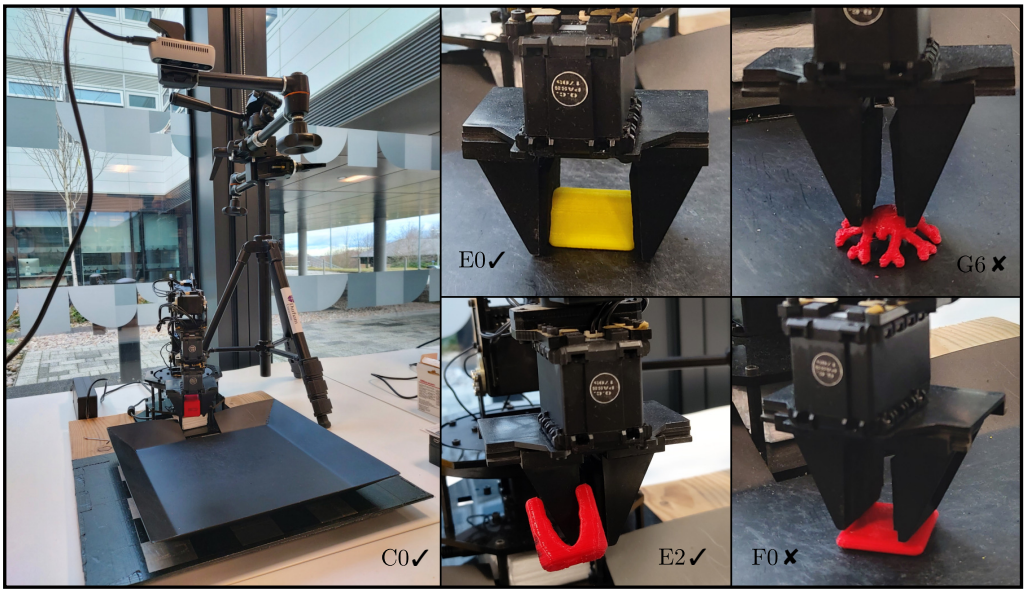


Figure 5.8: Example grasps on using a low-cost robotic arm with a parallel plate gripper. Most objects are grasped by reaching across the principal axis of the object (top left), however the model is also capable of plan grasps that only reached across parts of the object (bottom left). The most common failure cases are due to object difficulty where the model suggested grasps unsuitable for the type of gripper used (right images).

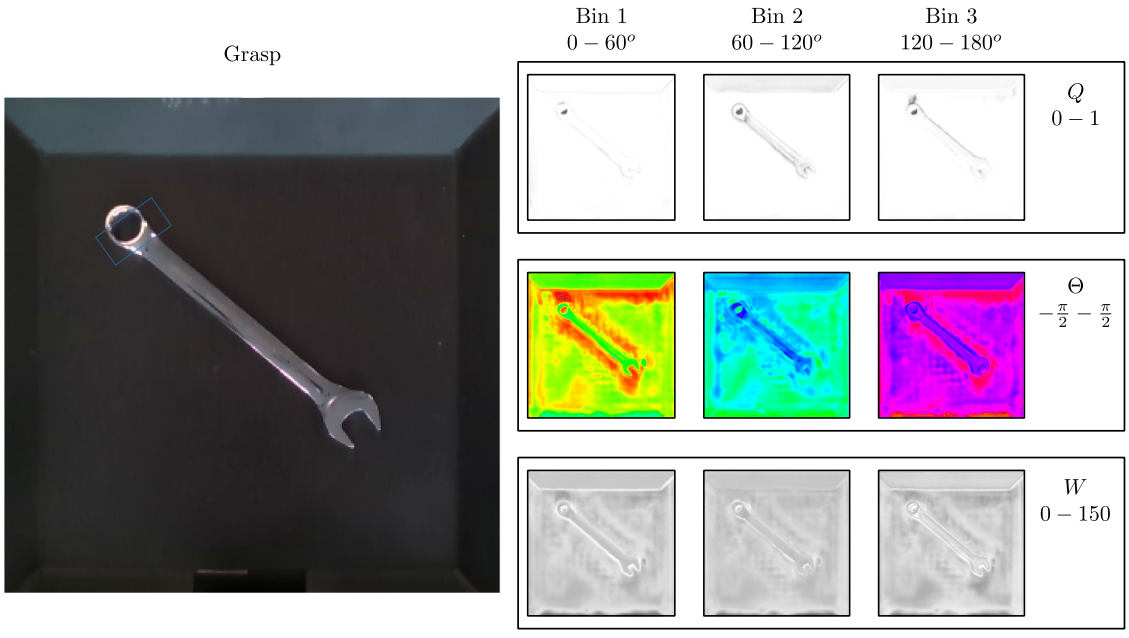


Figure 5.9: Example grasp alongside generative U-Net model [13] outputs when trained with multiple orientation bins and positional loss function.

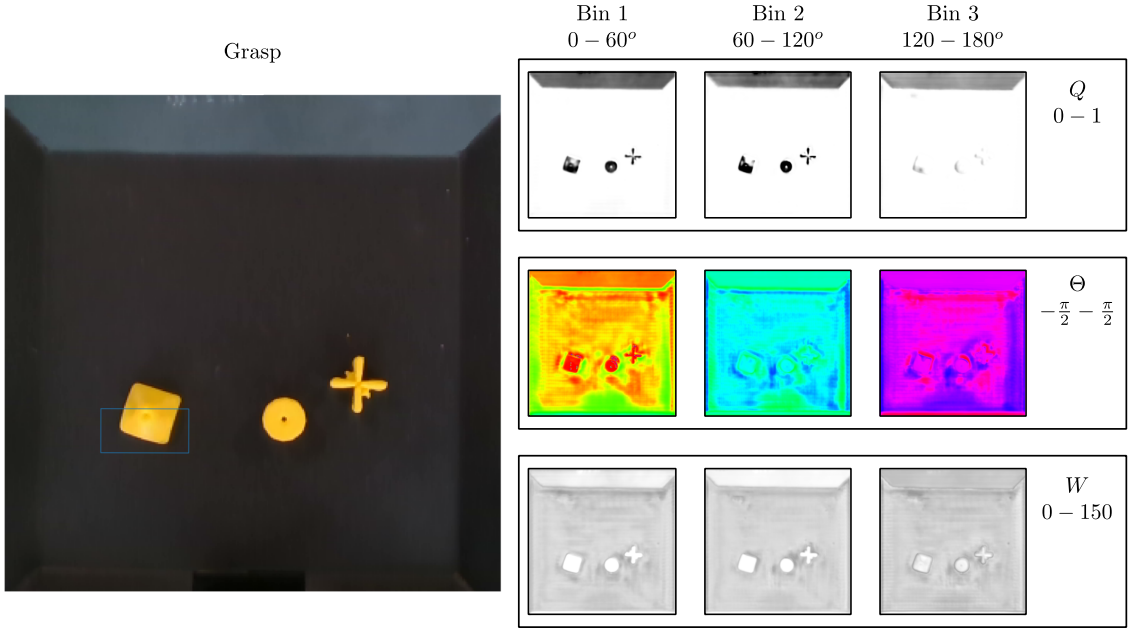


Figure 5.10: Example grasp alongside generative U-Net model [13] outputs when trained with multiple orientation bins and positional loss function. During implementation, the model is capable of suggesting grasps between multiple objects even though it was not explicitly trained to do so.

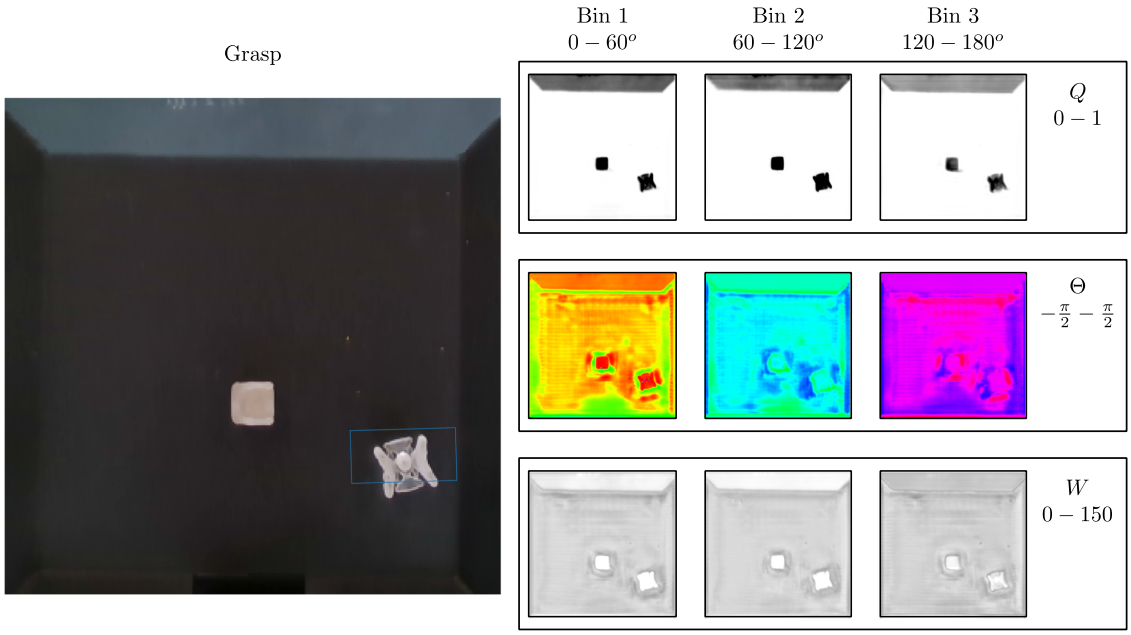


Figure 5.11: Example grasp alongside generative U-Net model [13] outputs when trained with multiple orientation bins and positional loss function. The model is also capable of distinguishing and suggesting grasps for translucent objects.

Table 5.3: Inference time and number of parameters of different grasping models.

Model	No. Parameters	Inference Time (ms)
Levine [20]	1M	200-500
GQ-CNN [10]	18M	800
FC-GQ-CNN [89]	-	625
GGCNN2	72k	12-14
GR-ConvNet	1.9M	38-40
U-Net	14.7M	25-28

a high success rate and operates at a much faster overall speed than other discriminative models, as shown in Table. 5.3. Our values are collected using GPU-acceleration on a NVidia GTX 1080Ti graphics card using *PyTorch* 1.1 with CUDA 11.

Whilst this robotic implementation requires an external depth camera, very few examples are failures as a result of an incorrect gripper depth. Future generative grasping models may benefit from an inherent depth module to directly predict (x, y, z) grasps without the need for the external transformation. Performance would likely be improved using more accurate robotic grippers and transfer to specific scenes or grippers (e.g. [187]). However, this work intends to minimise extensive retraining to reinforce the generalisability of the model.

5.3 Summary

This chapter evaluates the effectiveness of training common generative grasping models with a modified ground truth grasp map which applies a Gaussian filter. These results show that using both the attentional *positional loss* function, in addition to discrete orientation-specific outputs, together greatly improves performance.

Further experimentation has also shown the traditional rectangle metric, widely used as the most common measure of grasp performance, is insufficient for predicting grasp success on physical robot arms. These experiments show that models trained using a Gaussian ground truth, whilst showing negligible performance differences on the rectangle metric, were better able to propose appropriate grasps when testing on a simulated robot arm. Our best performing model achieves 87.94% grasp success according to the SGT, which is >2% performance increase over the previous state of the art on this benchmark [185].

We finally suggest the addition of the average intersection over union score in future work, as an offline metric for predicting real-world model performance. We also reinforce the need for testing of models on physical benchmarks in addition to offline measures and

therefore supplement this with real-world data to show the model is capable of transferring to the previously unseen physical EGAD object dataset [158]. The best model achieves high performance, even on complex, antagonist, and difficult to grasp objects without compromising on model inference speeds compared to larger discriminative grasping models.

CHAPTER 6

Conclusion

Due to the complex nature of creating robotic systems capable of fully autonomous grasping, there exists considerable amounts of literature aiming to generalise designed algorithms across a wide variety of objects and environments. This thesis set out to answer the main questions as set out in the introduction which outlined the desirable properties for a future system capable of dynamic object manipulation. A number of novel improvements were thus outlined for generative grasping models in the previous chapters. These class of generative models for robotic grasping were identified as a promising and reliable method for determining grasp quality because they are capable of grasp accuracy on par with the similar supervised discriminative models. These have the advantage that they enabled systems capable of closed-loop grasping due to their faster grasp inference times without the need of a separate ranking stage. One of the major limitations common across deep learning approaches for grasp estimation however, is generating large quantities of expertly labelled data for training and creating models which are efficient at learning the relevant grasping qualities.

The work presented here aims to address these issues by introducing changes within the training pipeline for these generative solutions to grasping. This included altering the preprocessing stage of grasp representation, the network architecture, and the loss function for training generative models. Furthermore, the methods for predicting and comparing offline model success on common datasets are evaluated and the real-world applicability

of these models is demonstrated when transferred to a robot arm. The following section outlines the main contributions of the thesis.

6.1 Contributions

Chapter 3 explores the use of multi-task learning in generative grasping architectures to improve the grasp accuracy from monocular inputs. As depth is an important aspect of 3D object geometry, grasp performance is significantly decreased in its absence. The aim was to improve the networks understanding of object geometry and subsequent grasp locations by training an end-to-end CNN architectures on simultaneous related tasks to learn common representations about each task in the early layers and improve primary grasp performance.

In Chapter 3, by using the early common Cornell Grasping Dataset (CGD), the first task examined was concurrent object classification, inspired by the dual-stream hypothesis of human object recognition and manipulation [25–27]. We classified each object into according to unique object classes or according to its functionality, with the idea that learning the best way to pick up an object is associated with its usage. However, alone this was not enough to improve grasp success rates. Forcing the early layers to learn tasks that relied on identifying different object properties, in fact negatively impacted performance. Decreasing the number of shared layers in this task improved this slightly, although it is likely that this was difficult for a relatively small number of layers and overall architecture, and the disparity between the regression and classification loss functions likely compounded this issue. Furthermore, it can be tedious to hand-label a grasping dataset with classifications which makes it unsuitable for larger datasets.

The rest of the chapter investigates an implementation of similar regression-based auxiliary tasks to match that of the primary grasping task. Without the requirement of classifying objects, instead the much larger and contemporary simulated Jacquard Grasping Dataset (JGD) was used for training. The network was trained to either perform salient object detection (SOD), using the corresponding object mask for the image, or depth reconstruction, using the perfect depth image included as part of the simulation. This time, both tasks were demonstrated to improve grasp success, with the best performing auxiliary SOD model improving grasp success from an average of 72.04 $\pm$3.44% to 78.84 $\pm$0.97%, however, the mean depth reconstruction performance was further improved to 79.50 $\pm$0.65%. This demonstrates that with carefully chosen tasks, multi-task

learning can improve performance when training end-to-end generative networks and even approaches performance of pure depth input 82.26 ±3.76%.

As a result of this work, Chapter 4 identified that these networks were relatively slow to train and required a large amount of data to reach peak performance. It was noted that a large proportion of the traditional loss function used for these networks was dedicated to learning about the irrelevant background information given only low top number of grasps are intended to be performed. We instead implemented a novel *positional loss* function which focuses learning towards suitable grasp locations. This generally improves predicted grasp success and learning stability on the JGD and a range of generative grasping architectures, achieving a total of 88.38±1.20% of successful grasps when using the GR-ConvNet, outperforming the traditional loss function. Crucially, this was also demonstrated to greatly improve performance when training on smaller datasets. When trained on the CGD, and tested on the same subset of data from the JGD, the positional loss function was shown to better generalise to the new setting and greater variety of objects which is a key problem. Furthermore, this improves performance with no impact on overall grasping inference times which means the model is still capable of dynamic control of an arm.

Finally, Chapter 5 recognises that the common IoU metric used to evaluate model performance on both the CGD and JGD suffers from a number of flaws. Initial implementations established an arbitrary threshold as a measure of defining offline grasp success. This can cause inflated and incorrect measures of success by ignoring collision data which may be crucial. Therefore, the necessity of a common simulated benchmark is reinforced for accurate comparisons between future models where collision data and model physics are taken into account. The SGT benchmark is suggested as it is easily accessible on a publicly accessible server using the same robotic equipment. Whilst physical benchmarks would be preferred, these can be costly to acquire and more difficult to set up.

This chapter goes on to introduce an alternative method for pre-processing and representing grasp rectangles. By using a Gaussian multiplier to alter ground truth grasp quality data, grasp centres are further emphasised without the need for more data collection. While there is likely to be some room for error in grasp placement, this method trains the network to generate grasps which are better centred against the ground truth. This is less likely to cause object collisions which consequently results in better performance according to the SGT metric, whereas the traditional metric does not accurately

predict this performance. Many studies tend to report their performance according to this offline metric so in future it is suggested that a corresponding mIoU metric should also be presented as a comparison metric. Both IoU, mIoU, and SGT performance also improves when trained with the positional loss function, demonstrating that this technique helps to generalise to other settings and environments.

Overall, the work presented in this thesis suggests generalised improvements for a series of generative grasping models and presents experiments according to their effectiveness. A final demonstration of the transferability of these models to a functional robot arm is made by testing the capability of the arm on the EGAD dataset [158], a previously unseen set of 3D printed objects of varying difficulty and complexity. Without additional data, the model is able to accurately predict grasp location, orientation, and width, which we believe demonstrates the practical applicability of the model by showing an ability to generalise to unseen objects and a totally new environment and background scene. Therefore, this addresses many of the research questions and limitations of the field as set out in Chapter 1.

6.2 Limitations and Future Work

Despite demonstrating promising performance for object manipulation with the use of these generative grasping models, there still exist a number of general limitations which need to be addressed for fully in the wider literature. This section discusses future directions for the field as well as concurrent work.

6.2.1 End-Effectors and Data Availability

The techniques outlined in this thesis are specifically designed for the grasp estimation and control of parallel plate grippers. While these are the common choice for many instances of robotic applications, such grippers have three major limitations in view of their ability to perform advanced autonomous tasks [39]:

- only objects with parallel and plane surfaces can be comfortably grasped, while objects with uneven surfaces of arbitrary shapes cannot be grasped easily;
- small reorientations of the object cannot be performed by the gripper alone, meaning the entire arm has to be moved, however fine adjustments are often difficult and time consuming to account for;

- structural properties of the grasped object, such as the surface texture, cannot be inferred via such grippers.

These properties are fine in industrial settings for their ability to perform pick and place tasks with high success rates and are relatively robust to damage, they are not necessarily appropriate for potential future domestic applications [1]. All three problems can be overcome by switching to end effectors capable of emulating human hand-like motions and sensing abilities, such as multi-fingered hands equipped with haptic feedback sensors like in prior work by [128, 188–190]. Robots equipped with these properties show promising results but multi-fingered grippers are more prone to damage and requires systems capable of planning with fine motor control. It has previously been mentioned that some systems employ dual-arm strategies to utilise the advantages of multiple gripper types, such as with a suction-based gripper in addition to a plate gripper [10]. Although current multi-fingered grippers allows for some levels of dexterous object manipulation [87], their usage is still very limited [191].

The main drawback of these approaches currently is the lack of available data for training for a variety of arms and end-effectors. For example, datasets do not typically contain multi-point finger placements. However, datasets are increasing in scope with examples such as GraspNet 1billion [192] including and orders of magnitude more grasps from multiple viewing angles of point cloud data, or RoboNet [193] which is an open source dataset for collating data from multiple arms and environments but more work is needed for creating easily accessible benchmarks to compare algorithms between. REPLAB [18] for example, created a cheap open source cell for testing and comparing algorithms in a standardised physical environment. Future research is encouraged to compare the results from this thesis with using the EGAD dataset [158] in the same methodology as Chapter 5.

6.2.2 Object Properties

Another limitation of the methods outlined so far is that they are trained via objects generated in simulation. Currently, the objects do not possess many of the properties which would be encountered in the wider world, as the objects are assumed to be completely rigid and opaque. In practicality, these systems must learn to also grasp objects which are currently difficult to grasp, including highly malleable objects such as cloth, which are difficult to grasp as they possess an infinite number of degrees of freedom [194], or transparent objects because they are often difficult to detect with depth sensors [195].

Understanding object properties such as material composition would help to predict the grasp required and react to changes in pressure.

6.2.3 Object Manipulation for Task Completion

When considering complex tasks, grasp synthesis alone cannot be considered an isolated problem. The research in this thesis is presented as parts of a wider goal with the intention of being implemented in the completion of higher level tasks and research has only just begun to investigate this phenomenon with the resources available.

While initially Chapter 3 investigates the use of concurrent grasping and classification in a multi-task system to tackle this problem, the difference in task representation ultimately created a trade-off between performance across both tasks separately unless they were highly related. Other research however, has also looked at simultaneous classification with grasping, although these methods typically employs a modular approaches instead of training end-to-end [196–198]. This may be more similar to the cortical regions of the visual system as each area is highly specialised for a given role, such as the fusiform face area (FFA) in the ventral stream [28] involved in face recognition, or anterior supramarginal gyrus (aSMG) for the recognition of tools which are distinct from general object recognition [155].

For practical applications, the type of grasp must therefore be considered for task completion. While it is easy to just create a gripper which would envelop an object for easy picking, this would not allow for goal-oriented task manipulation. Some work applies task constraints, like learning to classify grasps based on affordances [36, 199], with the intention that the system will, for example, grasp the handle of a hammer or a mug, which is crucial when tool functionality is task relevant. However, there is once again a lack of large-scale training data for this task. This is preferable to training robots which only aim to grasp what is in front of itself and opens up research for robots capable of higher level understanding and completion of multi-step tasks.

Some works are also starting to use reinforcement learning to manipulate objects prior to grasping in order to make it easier to grasp objects in clutter, such as pushing [35, 79], while others are starting to combine language models with simple tasks such as pick or place to complete orders based on intention of language understanding [200]. Future work would likely benefit from a holistic approach to grasping and integrate the approaches of supervised, unsupervised, and reinforcement learning methods. This would better mimic

the way toddlers develop grasping skills, which are not isolated from one another [201], and combine their advantages, including learning the types of grasps required for task completion, as well as learning to dexterous manipulation the object once grasped.

Bibliography

[1] J. Sanchez, J. A. Corrales, B. C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey," *International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018. 1, 1.1, 2, 2, 6.2.1

[2] A. Bicchi and V. Kumar, "Robotic Grasping and Contact: A Review," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 1, pp. 348 – 353, 2000. 1, 1.1, 2, 2.1

[3] S. Caldera, A. Rassau, and D. Chai, "Review of Deep Learning Methods in Robotic Grasp Detection," *Multimodal Technologies and Interaction*, vol. 2, pp. 1–24, 9 2018. 1, 2, 2.2, 2.3.1

[4] J. Bohg, A. Morales, T. Asfour, and D. Kragić, "Data-driven grasp synthesis - A survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014. 1, 1.1, 2, 2.1, 2.2, 2.2, 4

[5] D. Morrison, P. Corke, and J. Leitner, "Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach," *Robotics: Science and Systems*, pp. 1–10, 2018. 1, 2, 2.2, 2.1, 2.2.2, 2.3.1, 3, 3.1.2, 3.1.2, 3.1.3, 3.2.2, 4, 4, 4.1.3, 4.3, 5, 5.1.1, 5.3, 5.2, 5.2

[6] R. Hodson, "A gripping problem," *Nature*, vol. 557, pp. 523–525, 2018. 1.1

[7] D. Morrison, A. W. Tow, M. McTaggart, R. Smith, N. Kelly-Boxall, S. Wade-Mccue, J. Erskine, R. Grinover, A. Gurman, T. Hunn, D. Lee, A. Milan, T. Pham, G. Rallos, A. Razjigaev, T. Rowntree, K. Vijay, Z. Zhuang, C. Lehnert, I. Reid, P. Corke, and J. Leitner, "Cartman: The Low-Cost Cartesian Manipulator that Won the Amazon Robotics Challenge," in *Proc. IEEE International Conference on Robotics and Automation*, pp. 7757–7764, Institute of Electrical and Electronics Engineers Inc., 9 2018. 1.1, 2.2

[8] J. Leitner, "Picking the right robotics challenge," *Nature Machine Intelligence*, vol. 1, p. 162, 3 2019. 1.1

[9] E. Matsumoto, M. Saito, A. Kume, and J. Tan, "End-to-End Learning of Object Grasp Poses in the Amazon Robotics Challenge," *Advances on Robotic Item Picking*, pp. 63–72, 2020. 1.1

[10] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds Using a New Analytic Model and Deep Learning," in *Proc. IEEE International Conference on Robotics and Automation*, pp. 5620–5627, IEEE, 9 2018. 1.1, 2, 2.2, 2.1, 2.2.1, 2.3.1, 5.3, 6.2.1

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, p. 19, 2012. 1.1, 2.2, 3.1

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016. 1.1, 2, 2.2.2, 3.1

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, 2015. 1.1, 2, 5.1.1, 5.2, 5.1, 5.2, 5.9, 5.10, 5.11

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems*, pp. 241–294, 2017. 1.1

[15] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017. 1.1, 2, 2.2

[16] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A Survey on Learning-Based Robotic Grasping," *Current Robotics Reports*, vol. 1, pp. 239–249, 12 2020. 1.1, 1.1, 2.2, 2.2.2, 2.2.3, 2.3.3, 4

[17] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review," *Artificial Intelligence Review*, 2020. 1.1, 2.2

[18] B. Yang, D. Jayaraman, J. Zhang, and S. Levine, "REPLAB: A reproducible low-cost arm benchmark for robotic learning," in *Proc. IEEE International Conference on Robotics and Automation*, pp. 8691–8697, 5 2019. 1.1a, 2.3.2, 5.1.5, 6.2.1

[19] W. Prew, T. P. Breckon, M. Bordewich, and U. Beierholm, "Evaluating Gaussian Grasp Maps for Generative Grasping Models," in *Proc. IEEE International Joint Conference on Neural Networks*, 2022. 1.1a

[20] 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. 1.1b, 2.1, 2.2.3, 2.3, 2.3.2, 3, 3.1.1, 5.3

[21] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 3406–3413, 9 2016. 1.1c, 2.1, 2.3.1, 2.2, 2.3, 2.3.2, 2.3.2

[22] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, 1 2019. 1.1d, 2.2, 2.1, 2.2.1

[23] P. Gaussier and A. Pitti, "Reaching and Grasping : what we can learn from psychology and robotics," in *Reach-to-Grasp Behaviour*, pp. 349–366, Taylor & Francis Group, 2018. 1.1

[24] A. M. Dollar, L. P. Jentoft, J. H. Gao, and R. D. Howe, "Contact sensing and grasping performance of compliant hands," *Autonomous Robots*, vol. 28, no. 1, pp. 65–75, 2010. 1.1

[25] L. G. Ungerleider and M. Mishkin, "Two cortical visual systems," in *Analysis of Visual Behaviour*, MIT Press, 1982. 1.1, 2.2, 2.2.2, 3.1.2, 6.1

[26] M. A. Goodale and A. D. Milner, "Separate Visual Pathways for Perception and Action," *Trends in Neurosciences*, vol. 15, no. 1, pp. 20–25, 1992. 1.1, 2.2, 2.2.2, 3.1, 3.1.2, 3.1.3, 6.1

[27] A. D. Milner, "How do the two visual streams interact with each other?," *Experimental Brain Research*, vol. 235, no. 5, pp. 1297–1308, 2017. 1.1, 2.2, 2.2.2, 3.1, 3.1.2, 6.1

[28] N. Kanwisher, M. M. Chun, and J. McDermott, "The fusiform face area: a module in human extrastriate cortex specialized for face perception," *Journal of Neuroscience*, vol. 17, no. 11, pp. 4302–11, 1997. 1.1, 6.2.3

[29] M. A. Goodale, "Vision for perception and vision for action in the primate brain.," *Novartis Foundation Symposium*, vol. 218, pp. 21–34, 1998. 1.1

[30] R. A. Andersen, G. K. Essick, and R. M. Siegel, "Encoding of Spatial Location by Posterior Parietal Neurons," *Science*, vol. 230, no. 4724, pp. 456–458, 1985. 1.1

[31] D. Zipser and R. A. Andersen, "A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons," *Nature*, vol. 331, no. 6158, pp. 679–684, 1988. 1.1

[32] E. Salinas and P. Thier, "Gain modulation: A major computational principle of the central nervous system," in *Neuron*, 2000. 1.1

[33] S. Mahé, R. Braud, P. Gaussier, M. Quoy, and A. Pitti, "Exploiting the gain-modulation mechanism in parieto-motor neurons: Application to visuomotor transformations and embodied simulation," *Neural Networks*, vol. 62, pp. 102–111, 2 2015. 1.1

[34] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Object-based affordances detection with Convolutional Neural Networks and dense Conditional Random Fields," in *Proc. International Conference on Intelligent Robots and Systems*, pp. 5908–5915, Institute of Electrical and Electronics Engineers Inc., 12 2017. 1.1, 3

[35] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning Synergies between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning," *Proc. International Conference on Intelligent Robots and Systems*, pp. 4238–4245, 2018. 1.1, 2, 2.2, 6.2.3

[36] P. Ardon, E. Pairet, R. P. Petrick, S. Ramamoorthy, and K. S. Lohan, "Learning Grasp Affordance Reasoning through Semantic Relations," *IEEE Robotics and Automation Letters*, vol. 4, pp. 4571–4578, 6 2019. 1.1, 2.2.2, 2.3.1, 3, 3.1.3, 3.3, 6.2.3

[37] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016. 1.1, 2.2, 2.2.3, 2.3, 4.1.2

[38] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," *Proc. International Conference on Intelligent Robots and Systems*, vol. 2017-Septe, pp. 769–776, 2017. 1.4, 2, 2.2, 2.2.2, 2.3.1, 2.3.2, 3, 3.1.2

[39] K. B. Shimoga, "Robot grasp synthesis algorithms: A survey," *International Journal of Robotics Research*, vol. 15, no. 3, pp. 230–266, 1996. 2, 2.1, 2.2, 6.2.1

[40] Z. Ju, C. Yang, Z. Li, L. Cheng, and H. Ma, "Teleoperation of humanoid baxter robot using haptic feedback," in *Proc. International Conference on Multisensor Fusion and Information Integration for Intelligent Systems*, pp. 3–8, IEEE, 2014. 2

[41] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. International Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 779–788, 2016. 2, 2.2, 2.2.2

[42] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from RGBD images: Learning using a new rectangle representation," in *Proc. International Conference on Robotics and Automation*, pp. 3304–3311, IEEE, 2011. 2, 2.2, 2.2.2, 2.3.1, 2.5, 2.2, 2.3, 2.3.2, 3.1.2, 5, 5.1.2, 5.1.3, 5.2

[43] U. Asif, J. Tang, and S. Harrer, "Densely Supervised Grasp Detector (DSGD)," *Proc. AAAI Conference on Artificial Intelligence*, vol. 33, pp. 8085–8093, 2019. 2, 2.2.2, 2.3.2, 3, 4.1.1

[44] C. Finn, I. J. Goodfellow, and S. Levine, "Unsupervised Learning for Physical Interaction through Video Prediction," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2016. 2

[45] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent Models of Visual Attention," in *Proc. zhanAdvances in Neural Information Processing Systems*, vol. 27, pp. 1–9, 2014. 2

[46] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," in *Proc. Australasian Conference on Robotics and Automation*, 2015. 2, 2.3.1, 2.2

[47] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *International Journal of Robotics Research*, vol. 39, pp. 183–201, 3 2019. 2, 2.2.2, 3, 3.2.3, 4.1, 4.2.3, 4.3, 4.4, 5.1.1

[48] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight Experience Replay," in *Proc. Advances in Neural Information Processing Systems*, 2017. 2, 2.2.3, 2.3.3

[49] B. Siciliano, O. Khatib, and T. Kröger, *Springer Handbook of Robotics*. Berlin: Springer International Publishing, 1 2016. 2, 2.2.2, 4

[50] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3D object grasp synthesis algorithms," *Robotics and Autonomous Systems*, vol. 60, pp. 326–336, 3 2012. 2.1, 2.2

[51] R. M. Murray, Z. Li, and S. Shankar Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994. 2.1

[52] D. Prattichizzo and J. C. Trinkle, "Grasping," in *Springer Handbook of Robotics*, pp. 955–988, Cham: Springer, 2008. 2.1

[53] M. T. Mason and J. K. Salisbury Jr, *Robot Hands and the Mechanics of Manipulation*. MIT Press, 1985. 2.1

[54] V.-D. Nguyen, "Constructing Force-Closure Grasps," *International Journal of Robotics Research*, vol. 7, no. 3, pp. 3–16, 1988. 2.1

[55] J. Montana, David, "The Condition for Contact Grasp Stability," in *Proc. International Conference on Robotics and Automation*, 1991. 2.1

[56] J. C. Trinkle, "On the Stability and Instantaneous Velocity of Grasped Frictionless Objects," in *Proc. International Conference on Robotics and Automation*, pp. 560–572, 1992. 2.1

[57] A. Bicchi, "On the Closure Properties of Robotic Grasping," *The International Journal of Robotics Research*, vol. 14, no. 4, pp. 319–334, 1995. 2.1

[58] W. S. Howard and V. Kumar, "On the stability of grasped objects," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 6, pp. 904–917, 1996. 2.1

[59] T. Yoshikawa and K. Nagai, "Evaluation and Determination of Grasping Forces for Multi-fingered Hands," in *International Journal of Robotics Research*, pp. 2–5, 1988. 2.1

[60] Z. Li and S. S. Sastry, "Task-Oriented Optimal Grasping by Multifingered Robot Hands," *IEEE Journal on Robotics and Automation*, vol. 4, no. 1, pp. 32–44, 1988. 2.1

[61] Z. Li, P. Hsu, and S. Sastry, "Grasping and Coordinated Manipulation by a Multifingered Robot Hand," *The International Journal of Robotics Research*, vol. 8, no. 4, pp. 33–50, 1989. 2.1

[62] A. Rodriguez, M. T. Mason, and S. Ferry, "From caging to grasping," *International Journal of Robotics Research*, vol. 31, no. 7, pp. 886–900, 2012. 2.1

[63] C. Rosales, R. Suárez, M. Gabiccini, and A. Bicchi, "On the synthesis of feasible and prehensile robotic grasps," *Proc. International Conference on Robotics and Automation*, pp. 550–556, 2012. 2.1

[64] J. Seo, S. Kim, and V. Kumar, "Planar, Bimanual, Whole-Arm Grasping," in *Proc. International Conference on Robotics and Automation*, pp. 3271–3277, IEEE, 2012. 2.1

[65] A. Morales, T. Asfour, P. Azad, S. Knoop, and R. Dillmann, "Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands," in *Proc. International Conference on Intelligent Robots and Systems*, pp. 5663–5668, 2006. 2.2

[66] S. Ekvall and D. Kragic, "Learning and evaluation of the approach vector for automatic grasp generation and planning," in *Proc. International Conference on Robotics and Automation*, pp. 4715–4720, 2007. 2.2

[67] Y. h. Li, Q. j. Lei, C. Cheng, G. Zhang, W. Wang, and Z. Xu, "A review: Machine learning on robotic grasping," in *Proc. International Conference on Machine Vision*, p. 54, 2018. 2.2

[68] R. Liu, F. Nageotte, P. Zanne, M. de Mathelin, and B. Dresp-Langley, "Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review," *Robotics*, vol. 10, no. 1, pp. 1–13, 2021. 2.2, 2.2.3

[69] 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. 2.2, 2.2.1

[70] F. Spenrath and A. Pott, "Gripping Point Determination for Bin Picking Using Heuristic Search," in *Procedia CIRP*, vol. 62, pp. 606–611, Elsevier B.V., 1 2017. 2.2

[71] T. Hodăn, J. Matas, and . Obdržálek, "On evaluation of 6D object pose estimation," in *Proc. European Conference on Computer Vision*, pp. 609–619, Springer Verlag, 2016. 2.2

[72] R. Bregier, F. Devernay, L. Leyrit, and J. L. Crowley, "Symmetry Aware Evaluation of 3D Object Detection and Pose Estimation in Scenes of Many Parts in Bulk," in *Proc. International Conference on Computer Vision Workshops*, vol. 2018-Janua, pp. 2209–2218, 2017. 2.2

[73] F. Spenrath and A. Pott, "Using Neural Networks for Heuristic Grasp Planning in Random Bin Picking," in *Proc. International Conference on Automation Science and Engineering*, pp. 258–263, IEEE, 2018. 2.2

[74] A. Saxena, J. Driemeyer, J. Kearns, C. Osondu, and A. Y. Ng, "Learning to grasp novel objects using vision," *Springer Tracts in Advanced Robotics*, vol. 39, pp. 33–42, 2008. 2.2, 2.3.2, 2.3

[75] M. Adjigble, N. Marturi, V. Ortenzi, V. Rajasekaran, P. Corke, and R. Stolkin, "Model-free and learning-free grasping by Local Contact Moment matching," in *Proc. International Conference on Intelligent Robots and Systems*, pp. 2933–2940, Institute of Electrical and Electronics Engineers Inc., 12 2018. 2.2

[76] 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*, vol. 13, 3 2017. 2.2, 2.1, 2.2.1, 2.2.2, 2.3.1, 2.3, 2.3.2, 2.3.2

[77] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *Proc. International Conference on Robotics and Automation*, pp. 1316–1322, 2015. 2.2, 2.1, 2.2.2, 2.3.1, 2.3.1, 2.3.2, 3, 3.0.1, 3.1, 3.1.2, 4, 4.2, 5

[78] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015. 2.2, 2.1, 2.2.1, 2.3.1, 2.5, 2.2, 2.3.1, 2.3.2, 2.3, 2.3.2, 2.7, 3, 3.0.1, 3.1.1, 3.1, 3.1.2, 3.1.2, 3.2.1, 3.2.2, 3.2.2, 4.1.2, 4.1.3, 4.2.3, 5, 5.1.2, 5.1.3, 5.2

[79] L. Pinto and A. Gupta, "Learning to push by grasping: Using multiple tasks for effective learning," *Proc. IEEE International Conference on Robotics and Automation*, pp. 2161–2168, 9 2017. 2.2, 2.2.1, 6.2.3

[80] X. Zhou, X. Lan, H. Zhang, Z. Tian, Y. Zhang, and N. Zheng, "Fully Convolutional Grasp Detection Network with Oriented Anchor Box," *Proc. International Conference on Intelligent Robots and Systems*, pp. 7223–7230, 12 2018. 2.2, 2.2.2

[81] D. Park, Y. Seo, D. Shin, J. Choi, and S. Y. Chun, "A Single Multi-Task Deep Neural Network with Post-Processing for Object Detection with Reasoning and Robotic Grasp Detection," in *Proc. International Conference on Robotics and Automation*, pp. 7300–7306, 2020. 2.2

[82] M. Zhu and C. Fu, "Convolutional Neural Networks combined with Runge-Kutta Methods," *ArXiv*, pp. 1–10, 2018. 2.2

[83] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *Journal of Manufacturing Systems*, vol. 48, pp. 144–156, 2018. 2.2

[84] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "PVN3D: A deep point-wise 3D keypoints voting network for 6DoF pose estimation," in *Proc. Computer Vision and Pattern Recognition*, pp. 11629–11638, IEEE Computer Society, 2020. 2.2

[85] A. Mousavian, C. Eppner, and D. Fox, "6-DOF GraspNet: Variational grasp generation for object manipulation," in *Proc. International Conference on Computer Vision*, pp. 2901–2910, 5 2019. 2.2, 2.1

[86] B. Zhao, H. Zhang, X. Lan, H. Wang, Z. Tian, and N. Zheng, "REGNet: REgion-based Grasp Network for Single-shot Grasp Detection in Point Clouds," in *Proc. International Conference on Robotics and Automation*, pp. 13474–13480, 2 2021. 2.2

[87] M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020. 2.2, 6.2.1

[88] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation," in *Proc. Conference on Robotic Learning*, pp. 651–673, 2018. 2.1, 2.2.3

[89] V. Satish, J. Mahler, and K. Goldberg, "On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks," *IEEE Robotics and Automation Letters*, vol. 4, pp. 1357–1364, 4 2019. 2.1, 2.2.1, 5.3

[90] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks," in *Proc. Computer Vision and Pattern Recognition*, 2019. 2.1, 2.2.3, 2.3.2, 2.3.3

[91] D. Park and S. Y. Chun, "Classification based Grasp Detection using Spatial Transformer Network," in *Proc. Conference on Robotics Research*, 2018. 2.2.1

[92] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kroger, J. Kuffner, and K. Goldberg, "Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a Multi-Armed Bandit model with correlated rewards," in *Proc. International Conference on Robotics and Automation*, pp. 1957–1964, Institute of Electrical and Electronics Engineers Inc., 6 2016. 2.2.1, 2.2.2, 2.3, 3.1.1

[93] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Proc. Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014. 2.2.2, 2.3.3

[94] A. Depierre, E. Dellandréa, and L. Chen, "Optimizing correlated graspability score and grasp regression for better grasp prediction," *ArXiv*, pp. 1–7, 2 2020. 2.2.2

[95] D. Guo, F. Sun, H. Liu, T. Kong, B. Fang, and N. Xi, "A hybrid deep architecture for robotic grasp detection," in *Proc.International Conference on Robotics and Automation*, pp. 1609–1614, IEEE, 7 2017. 2.2.2, 2.3.2

[96] R. Caruana, "A Dozen Tricks with Multitask Learning," in *Neural networks: Tricks of the trade*, pp. 165–191, Springer, Berlin, Heidelberg, 1998. 2.2.2, 3

[97] S. Ruder, "An Overview of Multi-Task Learning in Deep Neural Networks," *ArXiv*, 6 2017. 2.2.2, 3

[98] R. Girshick, "Fast R-CNN," in *Proc. International Conference on Computer Vision*, pp. 1440–1448, 2015. 2.2.2

[99] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. International Conference on Machine Learning*, pp. 160–167, 2008. 2.2.2

[100] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," in *Proc. Association for Computational Linguistics*, pp. 4487–4496, 1 2020. 2.2.2

[101] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *Proc. International Conference on Acoustics, Speech and Signal Processing*, pp. 8599–8603, 10 2013. 2.2.2

[102] Z. Dong, S. Liu, T. Zhou, H. Cheng, L. Zeng, X. Yu, and H. Liu, "PPR-Net: Point-wise Pose Regression Network for Instance Segmentation and 6D Pose Estimation in Bin-picking Scenarios," in *Proc. Conference on Intelligent Robots and Systems*, pp. 1773–1780, Institute of Electrical and Electronics Engineers Inc., 11 2019. 2.2.2

[103] D. A. Klein, B. Illing, B. Gaspers, D. Schulz, and A. B. Cremers, "Hierarchical Salient Object Detection for Assisted Grasping," in *Proc. International Conference on Robotics and Automation*, pp. 1–15, 2017. 2.2.2, 3

[104] E. Jang, S. Vijayanarasimhan, P. Pastor, J. Ibarz, and S. Levine, "End-to-End Learning of Semantic Grasping," in *Proc. Conference on Robotic Learning*, pp. 119–132, 2017. 2.2.2, 3

[105] M. Rei, "Semi-supervised multitask learning for sequence labeling," in *Proc. Conference Association for Computational Linguistics*, vol. 1, pp. 2121–2130, 4 2017. 2.2.2

[106] J. Baxter, "A Model of Inductive Bias Learning," *Journal of Artificial Intelligence Research*, vol. 12, p. 149, 2000. 2.2.2, 3

[107] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics," in *Proc. Computer Vision and Pattern Recognition*, pp. 7482–7491, 2018. 2.2.2, 3.1.2, 3.1.2, 3.2.2, 3.2.3, 3.3

[108] D. Morrison, P. Corke, and J. Leitner, "Multi-view picking: Next-best-view reaching for improved grasping in clutter," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 8762–8768, 9 2019. 2.2.2

[109] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 2018. 2.2.3, 1, 2.3, 2.3.2

[110] C. Finn, T. Yu, J. Fu, P. Abbeel, and S. Levine, "Generalising Skills with Semi-Supervised Reinforcement Learning," in *Proc. International Conference on Learning Representation*, pp. 1–11, 2017. 2.2.3

[111] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. 2.2.3

[112] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Communications*, vol. 7, pp. 1–10, 2016. 2.2.3

[113] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, "Deep Reinforcement Learning for Vision-Based Robotic Grasping: A Simulated Comparative Evaluation of Off-Policy Methods," in *Proc. International Conference on Robotics and Automation*, pp. 6284–6291, 2018. 2.2.3

[114] C. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, pp. 279–292, 1992. 2.2.3

[115] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution Strategies as a Scalable Alternative to Reinforcement Learning," in *ArXiv*, 3 2017. 2.2.3

[116] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An Introduction to Deep Reinforcement Learning," *Foundations and Trends in Machine Learning*, vol. 11, pp. 219–354, 12 2018. 2.2.3, 2.3

[117] S. Levine, N. Wagener, and P. Abbeel, "Learning Contact-rich Manipulation Skills with Guided Policy Search," in *Proc. International Conference on Robotics and Automation*, pp. 156–163, 2015. 2.2.3

[118] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards," *ArXiv*, pp. 1–10, 2017. 2.2.3

[119] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming Exploration in Reinforcement Learning with Demonstrations," in *Proc. International Conference on Robotics and Automation*, pp. 6292–6299, 2018. 2.2.3

[120] S. Song, A. Zeng, J. Lee, and T. Funkhouser, "Grasping in the Wild: Learning 6DoF Closed-Loop Grasping from Low-Cost Demonstrations," *IEEE Robotics and Automation Letters*, vol. 5, pp. 4978–4985, 7 2020. 2.2.3

[121] B. Spector and S. Belongie, "Sample-Efficient Reinforcement Learning through Transfer and Architectural Priors," in *ArXiv*, 1 2018. 2.2.3

[122] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. International Conference on Machine Learning*, vol. 3, pp. 1856–1868, 2017. 2.2.3

[123] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-Shot Visual Imitation Learning via Meta-Learning," in *Proc. International Conference on Machine Learning*, pp. 1126–1135, 2017. 2.2.3

[124] M. Q. Mohammed, K. L. Chung, and C. S. Chyi, "Review of deep reinforcement learning-based object grasping: Techniques, open challenges, and recommendations," *IEEE Access*, vol. 8, pp. 178450–178481, 2020. 2.3

[125] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng, "Robotic Grasping of Novel Objects," in *Proc. Advances in Neural Information Processing Systems*, pp. 1209–1216, 2007. 2.3.1, 2.2

[126] A. B. Watson, "QUEST+: A general multidimensional Bayesian adaptive psychometric method," *Journal of Vision*, vol. 17, no. 3, p. 10, 2017. 2.3.1

[127] 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. 2.3.1, 2.2

[128] R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. H. Adelson, and S. Levine, "More than a feeling: Learning to grasp and regrasp using vision and touch," *IEEE Robotics and Automation Letters*, vol. 3, pp. 3300–3307, 10 2018. 2.2, 2.3.1, 6.2.1

[129] A. Depierre, E. Dellandrea, and L. Chen, "Jacquard: A Large Scale Dataset for Robotic Grasp Detection," in *Proc. International Conference on Intelligent Robots and Systems*, pp. 3511–3516, IEEE, 2018. 2.3.1, 2.3, 2.3.2, 2.3.2, 2.8, 3, 3.1.1, 3.1.2, 3.2.1, 3.2.2, 3.2.2, 3.2.2, 4, 4.1.2, 4.1.3, 4.3, 5, 5.1.2, 5.1.3, 5.2

[130] L. Y. Ku, E. Learned-Miller, and R. Grupen, "Associating Grasp Configurations with Hierarchical Features in Convolutional Neural Networks," in *Proc. International Conference on Intelligent Robots and Systems*, pp. 2434–2441, 2017. 2.3.1

[131] F. J. Chu, R. Xu, and P. A. Vela, "Real-world Multi-object, Multi-grasp Detection," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3355–3362, 2018. 2.3.1

[132] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008. 2.3

[133] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in Manipulation Research: Using the Yale-CMU-Berkeley Object and Model Set," *IEEE Robotics and Automation Magazine*, vol. 22, pp. 36–52, 9 2015. 2.3, 2.6, 2.3.2, 3.1.1

[134] 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. 2.3.2

[135] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, A. Ray, J. Schneider, P. Welinder, W. Zaremba, and P. Abbeel, "Domain randomization and generative models for robotic grasping," in *Proc. International Conference on Intelligent Robots and Systems*, pp. 3482–3489, 2017. 2.3.2, 2.3.3

[136] R. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman, "Never Stop Learning: The Effectiveness of Fine-Tuning in Robotic Reinforcement Learning," in *Proc. Conference on Robotic Learning*, 4 2020. 2.3.2

[137] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-CMU-Berkeley dataset for robotic manipulation research," *IEEE International Journal of Robotics Research*, vol. 36, pp. 261–268, 4 2017. 2.6, 2.3.2, 3.1.1

[138] A. X. Chang, T. Funkhouser, L. 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," 12 2015. 2.3.2

[139] M. Savva, A. X. Chang, and P. Hanrahan, "Semantically-Enriched 3D Models for Common-sense Knowledge," in *CVPR Workshop on Functionality, Physics, Intentionality and Causality*, 2015. 2.3.2

[140] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping," in *Proc. International Conference on Robotics and Automation*, pp. 4243–4250, Institute of Electrical and Electronics Engineers Inc., 9 2018. 2.3.3

[141] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *IEEE International Conference on Robotics and Automation*, pp. 3803–3810, Institute of Electrical and Electronics Engineers Inc., 9 2018. 2.3.3

[142] A. T. Miller and P. K. Allen, "Graspit: A versatile simulator for robotic grasping," *IEEE Robotics and Automation Magazine*, vol. 11, pp. 110–122, 12 2004. 2.3.3

[143] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016. 2.3.3

[144] B. O. Community, *Blender - a 3D modelling and rendering package*. Amsterdam: Stitching Blender Foundation, 2018. 2.3.3

[145] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2149–2154, 2004. 2.3.3

[146] S. James, A. J. Davison, and E. Johns, "Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task," in *Proc. Conference on Robotic Learning*, 2017. 2.3.3

[147] M. Danielczuk, M. Matl, S. Gupta, A. Li, A. Lee, J. Mahler, and K. Goldberg, "Segmenting unknown 3D objects from real depth images using mask R-CNN trained on synthetic data," in *Proc. International Conference on Robotics and Automation*, pp. 7283–7290, Institute of Electrical and Electronics Engineers Inc., 5 2019. 2.3.3

[148] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, V. Lempitsky, U. Dogan, M. Kloft, F. Orabona, T. Tommasi, and A. Ganin, "Domain-Adversarial Training of Neural Networks," *Journal of Machine Learning Research*, vol. 17, pp. 1–35, 2016. 2.3.3

[149] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proc. Computer Vision and Pattern Recognition*, pp. 95–104, 2017. 2.3.3

[150] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, "RL-CycleGAN: Reinforcement Learning Aware Simulation-To-Real," in *Proc. Computer Vision and Pattern Recognition*, pp. 11157–11166, 2020. 2.3.3

[151] D. Katz, A. Venkatraman, M. Kazemi, J. A. Bagnell, and A. Stentz, "Perceiving, learning, and exploiting object affordances for autonomous pile manipulation," *Autonomous Robots*, vol. 37, no. 4, pp. 369–382, 2014. 3

[152] H. Dang and P. K. Allen, "Semantic grasping: Planning task-specific stable robotic grasps," *Autonomous Robots*, vol. 37, no. 3, pp. 301–316, 2014. 3

[153] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes," in *Robotics: Science and Systems*, 11 2018. 3

[154] K. Hara, R. Vemulapalli, and R. Chellappa, "Designing Deep Convolutional Neural Networks for Continuous Object Orientation Estimation," *ArXiv*, pp. 1–10, 2017. 3.0.1, 3.1.2, 3.2.2

[155] G. A. Orban and F. Caruana, "The neural basis of human tool use," *Frontiers in Psychology*, vol. 5, no. 310, pp. 1–12, 2014. 3.1, 3.1.3, 6.2.3

[156] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *Computer Vision and Pattern Recognition*, pp. 248–255, 3 2009. 3.1

[157] 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 *Proc. European Conference on Computer Vision*, pp. 740–755, 2014. 3.1, 3.1.2

[158] 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, pp. 4368–4375, 3 2020. 3.1.2, 4, 4.1.1, 4.1.3, 4.3, 5.1.1, 5.1.2, 5.1.5, 5.5, 5.1.5, 5.2, 5.1, 5.2.2, 5.2.2, 5.7, 5.3, 6.1, 6.2.1

[159] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proc. International Conference on Learning Representations*, vol. 4, pp. 1–13, 2016. 3.1.2, 3.1.3

[160] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," *Proc. International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, pp. 4461–4468, 2016. 3.1.2

[161] G. Rizzolatti and M. Matelli, "Two different streams form the dorsal visual system: Anatomy and functions," *Experimental Brain Research*, vol. 153, no. 2, pp. 146–157, 2003. 3.1.3

[162] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, . Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems*, pp. 5999–6009, 6 2017. 3.1.3

[163] A. J. Kersey, T. S. Clark, C. A. Lussier, B. Z. Mahon, and J. F. Cantlon, "Development of tool representations in the dorsal and ventral visual object processing pathways," *Cerebral Cortex*, vol. 26, pp. 3135–3145, 7 2016. 3.1.3

[164] C. Koch and S. Ullman, "Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry," *Human Neurobiology*, vol. 4, no. 4, pp. 115–141, 1987. 3.2.1

[165] V. B. Mountcastle, R. A. Andersen, and B. C. Motter, "The influence of attentive fixation upon the excitability of the light-sensitive neurons of the posterior parietal cortex.," *Journal of Neuroscience*, vol. 1, no. 11, pp. 1218–25, 1981. 3.2.1

[166] B. C. Motter and V. B. Mountcastle, "The functional properties of the light-sensitive neurons of the posterior parietal cortex studied in waking monkeys: foveal sparing and opponent vector organization.," *Journal of Neuroscience*, vol. 1, no. 1, pp. 3–26, 1981. 3.2.1

[167] J. R. Whitlock, "Posterior parietal cortex," *Current Biology*, vol. 27, no. 14, pp. R691–R695, 2017. 3.2.1

[168] Y. E. Cohen and R. A. Andersen, "A common reference frame for movement plans in the posterior parietal cortex," *Nature Reviews Neuroscience*, vol. 3, no. 7, pp. 553–562, 2002. 3.2.1

[169] J. D. Crawford, W. P. Medendorp, and J. J. Marotta, "Spatial Transformations for EyeHand Coordination," *Journal of Neurophysiology*, vol. 92, no. 1, pp. 10–19, 2004. 3.2.1

[170] A. Takemura, Y. Inoue, K. Kawano, C. Quaia, and F. A. Miles, "Single-Unit Activity in Cortical Area MST Associated With Disparity-Vergence Eye Movements: Evidence for Population Coding," *Journal of Neurophysiology*, vol. 85, pp. 2245–2266, 5 2001. 3.2.1

[171] D. P. Munoz and R. H. Wurtz, "Saccade-related activity in monkey superior colliculus. I. Characteristics of burst and buildup cells.," *Journal of Neurophysiology*, vol. 73, pp. 2313–33, 6 1995. 3.2.1

[172] L. Itti, C. Koch, and E. Niebur, "A Model of Saliency-Based Visual Attention for Rapid Scene Analysis," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998. 3.2.1

[173] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H. Y. Shum, "Learning to detect a salient object," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 22, pp. 353–367, 2007. 3.2.1

[174] H. Laga, "A Survey on Deep Learning Architectures for Image-based Depth Reconstruction," *ArXiv*, pp. 1–28, 2019. 3.2.1

[175] H. Xue, S. Zhang, and D. Cai, "Depth Image Inpainting: Improving Low Rank Matrix Completion with Low Gradient Regularization," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4311–4320, 2017. 3.2.2, 5.1.1

[176] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations*, pp. 1–15, 2015. 3.2.2, 4.1.3, 5.1.4

[177] Y. Pang, X. Zhao, L. Zhang, and H. Lu, "Multi-scale Interactive Network for Salient Object Detection," in *Proc. Computer Vision and Pattern Recognition*, pp. 9410–9419, 2020. 3.2.2, 3.2.2

[178] W. Prew, T. Breckon, M. Bordewich, and U. Beierholm, "Improving Robotic Grasping on Monocular Images Via Multi-Task Learning and Positional Loss," in *Proc. IEEE International Conference on Pattern Recognition*, pp. 9843–9850, 2020. 4, 4.4, 5.2.1

[179] S. Kumra, S. Joshi, and F. Sahin, "Antipodal Robotic Grasping using Generative Residual Convolutional Neural Network," in *Proc. International Conference on Intelligent Robots and Systems*, pp. 1–8, 9 2020. 4, 4.1.1, 4.1.1, 4.2, 4.1.2, 4.1.3, 4.2.2, 4.2, 4.2.3, 4.3, 4.4, 4.3, 5.1.1, 5.1.1, 5.1.2, 5.1.4, 5.2, 5.1, 5.2

[180] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. 4.1.1

[181] U. Asif, J. Tang, and S. Harrer, "GraspNet: An efficient convolutional neural network for real-time grasp detection for low-powered devices," in *Proc. International Joint Conference on Artificial Intelligence*, pp. 4875–4882, 2018. 4.1.1

[182] U. Asif, J. Tang, and S. Harrer, "Ensemblenet: Improving Grasp Detection using an Ensemble of Convolutional Neural Networks," in *Proc. British Machine Vision Conference*, pp. 1–12, 2018. 4.1.1

[183] P. J. Huber, "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964. 4.1.2

[184] G. Chalvatzaki, P. Maragos, J. Peters, and N. Gkanatsios, "Revisiting Grasp Map Representation with a Focus on Orientation in Grasp Synthesis," in *Robotics: Science and Systems Workshop on Visual Learning and Reasoning for Robotic Manipulation*, pp. 1–5, 2020. 5, 5.1.1, 5.1.1, 5.3, 5.1.2, 5.1.4, 5.1.4, 5.2, 5.2.1

[185] A. Depierre, E. Dellandréa, and L. Chen, "Scoring Graspability based on Grasp Regression for Better Grasp Prediction," in *Proc. International Conference on Robotics and Automation*, pp. 4370–4376, 2021. 5.1.3, 5.2.1, 5.3

[186] G. Chalvatzaki, N. Gkanatsios, P. Maragos, and J. Peters, "Orientation Attentive Robot Grasp Synthesis," *ArXiv*, pp. 1–8, 2020. 5.1.4, 5.2.1

[187] K. Kleeberger, M. Völk, M. Moosmann, E. Thiessenhusen, F. Roth, R. Bormann, and M. F. Huber, "Transferring Experience from Simulation to the Real World for Precise Pick-And-Place Tasks in Highly Cluttered Scenes," in *IProc. International Conference on Intelligent Robots and Systems*, 2020. 5.2.2

[188] P. Falco, S. Lu, C. Natale, S. Pirozzi, and D. Lee, "A Transfer Learning Approach to Cross-Modal Object Recognition: From Visual Observation to Robotic Haptic Exploration," *IEEE Transactions on Robotics*, vol. 35, pp. 987–998, 8 2019. 6.2.1

[189] M. A. Lee, Y. Zhu, Y. Zhu, P. Zachares, M. Tan, K. Srinivasan, S. Savarese, L. Fei-Fei, A. Garg, A. Garg, and J. Bohg, "Making Sense of Vision and Touch: Learning Multimodal Representations for Contact-Rich Tasks," *IEEE Transactions on Robotics*, vol. 36, pp. 582–596, 6 2020. 6.2.1

[190] F. R. Hogan, J. Ballester, S. Dong, and A. Rodriguez, "Tactile Dexterity: Manipulation Primitives with Tactile Feedback," in *Proc. International Conference on Robotics and Automation*, pp. 8863–8869, Institute of Electrical and Electronics Engineers Inc., 2 2020. 6.2.1

[191] C. Yu and P. Wang, "Dexterous Manipulation for Multi-Fingered Robotic Hands With Reinforcement Learning: A Review," *Frontiers in Neurorobotics*, vol. 16, p. 861825, 2022. 6.2.1

[192] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "GraspNet-1Billion : A Large-Scale Benchmark for General Object Grasping," in *Proc. Conference on Vision and Pattern Recognition*, 2020. 6.2.1

[193] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, "RoboNet: Large-Scale Multi-Robot Learning," in *Proc. Conference on Robotic Learning*, pp. 1–13, 10 2019. 6.2.1

[194] J. Borràs, G. Alenyà, and C. Torras, "A Grasping-Centered Analysis for Cloth Manipulation," *IEEE Transactions on Robotics*, vol. 36, pp. 924–936, 6 2020. 6.2.2

[195] Y. Sun, J. Falco, M. A. Roa, and B. Calli, "Research Challenges and Progress in Robotic Grasping and Manipulation Competitions," *IEEE Robotics and Automation Letters*, vol. 7, pp. 874–881, 4 2022. 6.2.2

[196] M. Antonelli, E. Chinellato, and A. P. Del Pobil, "On-Line Learning of the Visuomotor Transformations on a Humanoid Robot," in *Intelligent Autonomous Systems* (S. Lee, H. Cho, K.-J. Yoon, and J. Lee, eds.), pp. 853–861, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. 6.2.3

[197] M. Antonelli, A. Gibaldi, F. Beuth, A. J. Duran, A. Canessa, M. Chessa, F. Solari, A. P. Del Pobil, F. Hamker, E. Chinellato, and S. P. Sabatini, "A hierarchical system for a distributed representation of the peripersonal space of a humanoid robot," *IEEE Transactions on Autonomous Mental Development*, vol. 6, no. 4, pp. 259–273, 2014. 6.2.3

[198] M. Antonelli, M. Rucci, and B. Shi, "Unsupervised learning of depth during coordinated head/eye movements," *Proc. International Conference on Intelligent Robots and Systems*, pp. 5199–5204, 2016. 6.2.3

[199] H. Dang and P. K. Allen, "Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task," in *Proc. International Conference on Intelligent Robots and Systems*, pp. 1311–1317, 2012. 6.2.3

[200] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, R. At Google, and E. Robots, "Do As I Can, Not As I Say: Grounding Language In Robotic Affordances," *ArXiv*, pp. 1–33, 2022. 6.2.3

[201] J. Atkinson and O. Braddick, "Visual Development," in *Oxford Handbook of Developmental Psychology Vol. 1: Body and Mind* (P. D. Zelazo, ed.), pp. 271–309, Oxford University Press, 2013. 6.2.3

APPENDIX A

Classification Labels for the Cornell Grasping Dataset

A.1 List of Specific Classification Labels

Table A.1: List of *specific* classification labels for the Cornell Grasping Dataset.

Label	Images
apple	pcd0820r, pcd0821r, pcd0822r, pcd0823r
asparagus	pcd0804r, pcd0805r, pcd0806r, pcd0807r
bag	pcd0280r, pcd0281r, pcd0282r, pcd0283r
ball	pcd0795r, pcd0796r
banana	pcd0848r, pcd0849r, pcd0850r, pcd0851r, pcd0852r, pcd0853r, pcd0854r, pcd0855r, pcd0856r, pcd0857r, pcd0858r, pcd0859r
baseball hat	pcd0876r, pcd0877r, pcd0878r, pcd0879r, pcd0880r, pcd0881r, pcd0882r, pcd0883r, pcd0884r, pcd0885r, pcd0886r, pcd0887r, pcd0888r, pcd0889r, pcd0890r, pcd0891r
book	pcd0622r, pcd0623r, pcd0624r, pcd0625r, pcd0626r, pcd0627r, pcd0628r, pcd0629r, pcd0630r, pcd0631r, pcd0632r
bowl	pcd0324r, pcd0325r, pcd0326r, pcd0801r, pcd0802r, pcd0803r

Label	Images
box	pcd0137r, pcd0138r, pcd0139r, pcd0140r, pcd0141r, pcd0142r, pcd0143r, pcd0144r, pcd0145r, pcd0146r, pcd0147r, pcd0148r, pcd0149r, pcd0150r, pcd0151r, pcd0152r, pcd0260r, pcd0261r, pcd0262r, pcd0263r, pcd0343r, pcd0344r, pcd0345r, pcd0346r, pcd0529r, pcd0530r, pcd0531r, pcd0532r
cable	pcd0375r, pcd0376r, pcd0377r, pcd0378r, pcd0379r, pcd0380r, pcd0381r, pcd0382r, pcd0383r, pcd0384r, pcd0385r, pcd0386r, pcd0475r, pcd0476r, pcd0477r, pcd0478r
calculator	pcd0256r, pcd0257r, pcd0258r, pcd0259r, pcd0351r, pcd0352r, pcd0353r, pcd0354r, pcd0723r, pcd0724r, pcd0725r, pcd0726r
camera	pcd0197r, pcd0198r, pcd0199r, pcd0200r, pcd0542r, pcd0543r, pcd0544r, pcd0545r
can	pcd0594r, pcd0595r, pcd0596r, pcd0597r, pcd0598r, pcd0599r, pcd0600r, pcd0601r, pcd0602r, pcd0603r, pcd0604r, pcd0605r
can opener	pcd0423r, pcd0424r, pcd0425r, pcd0426r, pcd0427r, pcd0428r, pcd0429r, pcd0430r
candle	pcd0447r, pcd0448r, pcd0449r, pcd0450r, pcd0451r, pcd0452r, pcd0453r, pcd0454r, pcd0455r, pcd0456r, pcd0457r, pcd0458r, pcd0459r, pcd0460r, pcd0461r, pcd0462r
capo	pcd0193r, pcd0194r, pcd0195r, pcd0196r, pcd0755r, pcd0756r, pcd0757r, pcd0758r
clip	pcd0439r, pcd0440r, pcd0441r, pcd0442r
corkscrew	pcd0431r, pcd0432r, pcd0433r, pcd0434r, pcd0435r, pcd0436r, pcd0437r, pcd0438r
courgette	pcd0864r, pcd0865r, pcd0866r, pcd0867r
cup	pcd0312r, pcd0313r, pcd0314r, pcd0315r, pcd0316r, pcd0317r, pcd0318r, pcd0319r, pcd0669r, pcd0670r, pcd0671r, pcd0672r, pcd0673r, pcd0674r, pcd0675r, pcd0676r, pcd0677r
deodorant	pcd0169r, pcd0170r, pcd0171r, pcd0172r, pcd0239r, pcd0240r, pcd0241r, pcd0242r, pcd0743r, pcd0744r, pcd0745r, pcd0746r

Label	Images
flip flop	pcd0633r, pcd0634r, pcd0635r, pcd0636r, pcd0637r, pcd0638r, pcd0639r, pcd0640r, pcd0641r, pcd0642r, pcd0643r, pcd0644r, pcd0645r, pcd0646r, pcd0647r, pcd0648r
floss	pcd0205r, pcd0206r, pcd0207r, pcd0208r
frisbee	pcd0247r, pcd0387r, pcd0388r, pcd0479r, pcd0480r
glasses	pcd0106r, pcd0107r, pcd0108r, pcd0109r, pcd0133r, pcd0134r, pcd0135r, pcd0136r, pcd0153r, pcd0154r, pcd0155r, pcd0156r, pcd0606r, pcd0607r, pcd0608r, pcd0609r, pcd0610r, pcd0611r, pcd0612r, pcd0613r, pcd0614r, pcd0615r, pcd0616r, pcd0617r, pcd0618r, pcd0619r, pcd0620r, pcd0621r
glasses case	pcd0161r, pcd0162r, pcd0163r, pcd0164r
goggles	pcd0185r, pcd0186r, pcd0187r, pcd0188r, pcd0189r, pcd0190r, pcd0191r, pcd0192r
hairbrush	pcd0165r, pcd0166r, pcd0167r, pcd0168r
headphones	pcd0327r, pcd0328r, pcd0329r, pcd0330r
kiwi	pcd0868r, pcd0869r, pcd0870r, pcd0871r
lemon	pcd0828r, pcd0829r, pcd0830r, pcd0831r
lightbulb	pcd0797r, pcd0798r, pcd0799r, pcd0800r, pcd1031r, pcd1032r
lime	pcd0824r, pcd0825r, pcd0826r, pcd0827r
lock	pcd0934r, pcd0935r, pcd0936r, pcd0937r, pcd0938r, pcd0939r, pcd0940r, pcd0941r, pcd0942r, pcd0943r, pcd0944r, pcd0945r, pcd0946r, pcd0947r, pcd0948r, pcd0949r, pcd1000r
lollipop	pcd1001r, pcd1002r, pcd1003r
mango	pcd0832r, pcd0833r, pcd0834r, pcd0835r
masher	pcd0759r, pcd0760r, pcd0761r, pcd0762r
mouse	pcd0122r, pcd0123r, pcd0124r, pcd0125r, pcd0443r, pcd0444r, pcd0445r, pcd0446r, pcd0517r, pcd0518r, pcd0519r, pcd0520r
mouthguard	pcd0217r, pcd0218r
mug	pcd0320r, pcd0321r, pcd0322r, pcd0323r, pcd0347r, pcd0348r, pcd0349r, pcd0350r, pcd0389r, pcd0390r, pcd0391r, pcd0392r, pcd0525r, pcd0526r, pcd0527r, pcd0528r

Label	Images
nail polish	pcd0918r, pcd0919r, pcd0920r, pcd0921r, pcd0922r, pcd0923r, pcd0924r, pcd0925r, pcd0926r, pcd0927r, pcd0928r, pcd0929r
onion	pcd0808r, pcd0809r, pcd0810r, pcd0811r, pcd0812r, pcd0813r, pcd0814r, pcd0815r
orange	pcd0335r, pcd0336r, pcd0337r, pcd0338r
peeler	pcd0419r, pcd0420r, pcd0421r, pcd0422r
pen	pcd0129r, pcd0130r, pcd0131r, pcd0132r, pcd0331r, pcd0332r, pcd0333r, pcd0334r, pcd0787r, pcd0788r, pcd0789r, pcd0790r, pcd0791r, pcd0792r, pcd0793r, pcd0794r, pcd1004r, pcd1005r, pcd1006r, pcd1007r, pcd1008r, pcd1009r, pcd1010r, pcd1011r, pcd1012r
pepper	pcd0836r, pcd0837r, pcd0838r, pcd0839r, pcd0840r, pcd0841r, pcd0842r, pcd0843r, pcd0844r, pcd0845r, pcd0846r, pcd0847r
pez	pcd0649r, pcd0650r, pcd0651r, pcd0652r, pcd0653r, pcd0654r, pcd0655r, pcd0656r, pcd0657r, pcd0658r, pcd0659r, pcd0660r, pcd0661r, pcd0662r, pcd0663r, pcd0664r, pcd0665r, pcd0666r, pcd0667r, pcd0668r
phone	pcd0296r, pcd0297r, pcd0298r, pcd0299r, pcd0300r, pcd0301r, pcd0302r, pcd0303r, pcd0304r, pcd0305r, pcd0306r, pcd0307r, pcd0308r, pcd0309r, pcd0310r, pcd0311r
plum	pcd0872r, pcd0873r, pcd0874r, pcd0875r
potato	pcd0860r, pcd0861r, pcd0862r, pcd0863r
pour bottle	pcd0463r, pcd0464r, pcd0465r, pcd0466r, pcd0467r, pcd0468r, pcd0469r, pcd0470r, pcd0471r, pcd0472r, pcd0473r, pcd0474r, pcd0930r, pcd0931r, pcd0932r, pcd0933r
razor	pcd0219r, pcd0220r, pcd0221r, pcd0222r, pcd0223r, pcd0224r, pcd0225r, pcd0226r, pcd0227r, pcd0228r, pcd0229r, pcd0230r, pcd0231r, pcd0232r, pcd0233r, pcd0234r, pcd0763r, pcd0764r, pcd0765r, pcd0766r, pcd0767r, pcd0768r, pcd0769r, pcd0770r
remote	pcd0100r, pcd0101r, pcd0489r, pcd0490r, pcd0491r, pcd0492r, pcd0493r, pcd0494r, pcd0495r, pcd0496r, pcd0501r, pcd0502r, pcd0503r, pcd0504r

Label	Images
ribbon	pcd0521r, pcd0522r, pcd0523r, pcd0524r, pcd0678r, pcd0679r, pcd0680r, pcd0681r, pcd0682r, pcd0683r, pcd0684r, pcd0685r, pcd0686r
rolling pin	pcd0485r, pcd0486r, pcd0487r, pcd0488r, pcd0715r, pcd0716r, pcd0717r, pcd0718r
scissors	pcd0248r, pcd0249r, pcd0250r, pcd0251r, pcd0513r, pcd0514r, pcd0515r, pcd0516r, pcd0546r, pcd0547r, pcd0548r, pcd0549r, pcd0550r, pcd0551r, pcd0552r, pcd0553r, pcd0554r, pcd0555r, pcd0556r, pcd0557r, pcd0558r, pcd0559r, pcd0560r, pcd0561r, pcd0687r, pcd0688r, pcd0689r, pcd0690r
screwdriver	pcd0578r, pcd0579r, pcd0580r, pcd0581r, pcd0582r, pcd0583r, pcd0584r, pcd0585r, pcd0586r, pcd0587r, pcd0588r, pcd0589r
shoe	pcd0110r, pcd0111r, pcd0112r, pcd0113r, pcd0114r, pcd0115r, pcd0116r, pcd0117r, pcd0264r, pcd0265r, pcd0266r, pcd0267r, pcd0268r, pcd0269r, pcd0270r, pcd0271r, pcd0272r, pcd0273r, pcd0274r, pcd0275r, pcd0276r, pcd0277r, pcd0278r, pcd0279r
soap	pcd0339r, pcd0340r, pcd0341r, pcd0342r
spatula	pcd0407r, pcd0408r, pcd0409r, pcd0410r, pcd0411r, pcd0412r, pcd0413r, pcd0414r, pcd0415r, pcd0416r, pcd0417r, pcd0418r
spice	pcd0355r, pcd0356r, pcd0357r, pcd0358r, pcd0359r, pcd0360r, pcd0361r, pcd0362r, pcd0363r, pcd0364r, pcd0365r, pcd0366r, pcd0367r, pcd0368r, pcd0369r, pcd0370r, pcd0371r, pcd0372r, pcd0373r, pcd0374r
sponge	pcd0562r, pcd0563r, pcd0564r, pcd0565r, pcd0566r, pcd0567r, pcd0568r, pcd0569r, pcd0570r, pcd0571r, pcd0572r, pcd0573r, pcd0574r, pcd0575r, pcd0576r, pcd0577r
spoon	pcd0399r, pcd0400r, pcd0401r, pcd0402r, pcd0403r, pcd0404r, pcd0405r, pcd0406r, pcd1033r, pcd1034r
spray bottle	pcd0292r, pcd0293r, pcd0294r, pcd0295r

Label	Images
squirt bottle	pcd0243r, pcd0244r, pcd0245r, pcd0246r, pcd0252r, pcd0253r, pcd0254r, pcd0255r, pcd0288r, pcd0289r, pcd0290r, pcd0291r, pcd0497r, pcd0498r, pcd0499r, pcd0500r, pcd0747r, pcd0748r, pcd0749r, pcd0750r, pcd0751r, pcd0752r, pcd0753r,pcd0754r, pcd0898r, pcd0899r, pcd0900r, pcd0901r, pcd0902r, pcd0903r, pcd0904r, pcd0905r, pcd0906r, pcd0907r, pcd0908r, pcd0909r, pcd0910r, pcd0911r, pcd0912r, pcd0913r, pcd0914r, pcd0915r, pcd0916r, pcd0917r
stapler	pcd0102r, pcd0103r, pcd0104r, pcd0105r, pcd0590r, pcd0591r, pcd0592r, pcd0593r
straw hat	pcd0892r, pcd0893r, pcd0894r, pcd0895r, pcd0896r, pcd0897r
string	pcd1013r, pcd1014r, pcd1015r, pcd1016r, pcd1017r, pcd1018r, pcd1019r, pcd1020r, pcd1021r, pcd1022r, pcd1023r, pcd1024r
sweet	pcd0691r, pcd0692r, pcd0693r, pcd0694r, pcd0695r, pcd0696r, pcd0697r, pcd0698r, pcd0699r, pcd0700r, pcd0701r, pcd0702r, pcd0703r, pcd0704r, pcd0705r, pcd0706r, pcd0707r, pcd0708r, pcd0709r, pcd0710r, pcd0711r, pcd0712r, pcd0713r, pcd0714r
tape	pcd0126r, pcd0127r, pcd0128r, pcd0393r, pcd0394r, pcd0395r, pcd0396r, pcd0397r, pcd0398r, pcd0533r, pcd0534r, pcd0535r, pcd0536r, pcd0537r, pcd0538r, pcd0539r, pcd0540r, pcd0541r
tennis balls	pcd0481r, pcd0482r, pcd0483r, pcd0484r
thread	pcd1025r, pcd1026r, pcd1027r, pcd1028r, pcd1029r, pcd1030r
toilet brush	pcd0783r, pcd0784r, pcd0785r, pcd0786r
tomato	pcd0816r, pcd0817r, pcd0818r, pcd0819r
toothbrush	pcd0173r, pcd0174r, pcd0175r, pcd0176r, pcd0177r, pcd0178r, pcd0179r, pcd0180r, pcd0213r, pcd0214r, pcd0215r, pcd0216r, pcd0235r, pcd0236r, pcd0237r, pcd0238r, pcd0727r, pcd0728r, pcd0729r, pcd0730r, pcd0771r, pcd0772r, pcd0773r, pcd0774r
toothpaste	pcd0181r, pcd0182r, pcd0183r, pcd0184r, pcd0201r, pcd0202r, pcd0203r, pcd0204r, pcd0209r, pcd0210r, pcd0211r, pcd0212r, pcd0775r, pcd0776r, pcd0777r, pcd0778r

Label	Images
torch	pcd0118r, pcd0119r, pcd0120r, pcd0121r, pcd0157r, pcd0158r, pcd0159r, pcd0160r, pcd0284r, pcd0285r, pcd0286r, pcd0287r, pcd0505r, pcd0506r, pcd0507r, pcd0508r, pcd0509r, pcd0510r, pcd0511r, pcd0512r
umbrella	pcd0731r, pcd0732r, pcd0733r, pcd0734r, pcd0735r, pcd0736r, pcd0737r, pcd0738r, pcd0739r, pcd0740r, pcd0741r, pcd0742r
whisk	pcd0719r, pcd0720r, pcd0721r, pcd0722r
wiper	pcd0779r, pcd0780r, pcd0781r, pcd0782r

A.2 List of General Classification Labels

Table A.2: List of *general* classification labels for the Cornell Grasping Dataset.

Label	Images
accessory	pcd0106r, pcd0107r, pcd0108r, pcd0109r, pcd0110r, pcd0111r, pcd0112r, pcd0113r, pcd0114r, pcd0115r, pcd0116r, pcd0117r, pcd0133r, pcd0134r, pcd0135r, pcd0136r, pcd0153r, pcd0154r, pcd0155r, pcd0156r, pcd0185r, pcd0186r, pcd0187r, pcd0188r, pcd0189r, pcd0190r, pcd0191r, pcd0192r, pcd0264r, pcd0265r, pcd0266r, pcd0267r, pcd0268r, pcd0269r, pcd0270r, pcd0271r, pcd0272r, pcd0273r, pcd0274r, pcd0275r, pcd0276r, pcd0277r, pcd0278r, pcd0279r, pcd0280r, pcd0281r, pcd0282r, pcd0283r, pcd0606r, pcd0607r, pcd0608r, pcd0609r, pcd0610r, pcd0611r, pcd0612r, pcd0613r, pcd0614r, pcd0615r, pcd0616r, pcd0617r, pcd0618r, pcd0619r, pcd0620r, pcd0621r, pcd0633r, pcd0634r, pcd0635r, pcd0636r, pcd0637r, pcd0638r, pcd0639r, pcd0640r, pcd0641r, pcd0642r, pcd0643r, pcd0644r, pcd0645r, pcd0646r, pcd0647r, pcd0648r, pcd0876r, pcd0877r, pcd0878r, pcd0879r, pcd0880r, pcd0881r, pcd0882r, pcd0883r, pcd0884r, pcd0885r, pcd0886r, pcd0887r, pcd0888r, pcd0889r, pcd0890r, pcd0891r, pcd0892r, pcd0893r, pcd0894r, pcd0895r, pcd0896r, pcd0897r

Label	Images
cleaning tool	pcd0173r, pcd0174r, pcd0175r, pcd0176r, pcd0177r, pcd0178r, pcd0179r, pcd0180r, pcd0181r, pcd0182r, pcd0183r, pcd0184r, pcd0201r, pcd0202r, pcd0203r, pcd0204r, pcd0205r, pcd0206r, pcd0207r, pcd0208r, pcd0209r, pcd0210r, pcd0211r, pcd0212r, pcd0213r, pcd0214r, pcd0215r, pcd0216r, pcd0219r, pcd0220r, pcd0221r, pcd0222r, pcd0223r, pcd0224r, pcd0225r, pcd0226r, pcd0227r, pcd0228r, pcd0229r, pcd0230r, pcd0231r, pcd0232r, pcd0233r, pcd0234r, pcd0235r, pcd0236r, pcd0237r, pcd0238r, pcd0339r, pcd0340r, pcd0341r, pcd0342r, pcd0562r, pcd0563r, pcd0564r, pcd0565r, pcd0566r, pcd0567r, pcd0568r, pcd0569r, pcd0570r, pcd0571r, pcd0572r, pcd0573r, pcd0574r, pcd0575r, pcd0576r, pcd0577r, pcd0727r, pcd0728r, pcd0729r, pcd0730r, pcd0763r, pcd0764r, pcd0765r, pcd0766r, pcd0767r, pcd0768r, pcd0769r, pcd0770r, pcd0771r, pcd0772r, pcd0773r, pcd0774r, pcd0775r, pcd0776r, pcd0777r, pcd0778r, pcd0779r, pcd0780r, pcd0781r, pcd0782r, pcd0783r, pcd0784r, pcd0785r, pcd0786r
container	pcd0137r, pcd0138r, pcd0139r, pcd0140r, pcd0141r, pcd0142r, pcd0143r, pcd0144r, pcd0145r, pcd0146r, pcd0147r, pcd0148r, pcd0149r, pcd0150r, pcd0151r, pcd0152r, pcd0161r, pcd0162r, pcd0163r, pcd0164r, pcd0169r, pcd0170r, pcd0171r, pcd0172r, pcd0217r, pcd0218r, pcd0239r, pcd0240r, pcd0241r, pcd0242r, pcd0243r, pcd0244r, pcd0245r, pcd0246r, pcd0252r, pcd0253r, pcd0254r, pcd0255r, pcd0260r, pcd0261r, pcd0262r, pcd0263r, pcd0288r, pcd0289r, pcd0290r, pcd0291r, pcd0292r, pcd0293r, pcd0294r, pcd0295r, pcd0343r, pcd0344r, pcd0345r, pcd0346r, pcd0355r, pcd0356r, pcd0357r, pcd0358r, pcd0359r, pcd0360r, pcd0361r, pcd0362r, pcd0363r, pcd0364r, pcd0365r, pcd0366r, pcd0367r, pcd0368r, pcd0369r, pcd0370r, pcd0371r, pcd0372r, pcd0373r, pcd0374r, pcd0447r, pcd0448r, pcd0449r, pcd0450r, pcd0451r, pcd0452r, pcd0453r, pcd0454r, pcd0455r, pcd0456r, pcd0457r, pcd0458r, pcd0459r, pcd0460r, pcd0461r, pcd0462r,

Label	Images
container	pcd0463r, pcd0464r, pcd0465r, pcd0466r, pcd0467r, pcd0468r, pcd0469r, pcd0470r, pcd0471r, pcd0472r, pcd0473r, pcd0474r, pcd0481r, pcd0482r, pcd0483r, pcd0484r, pcd0497r, pcd0498r, pcd0499r, pcd0500r, pcd0529r, pcd0530r, pcd0531r, pcd0532r, pcd0594r, pcd0595r, pcd0596r, pcd0597r, pcd0598r, pcd0599r, pcd0600r, pcd0601r, pcd0602r, pcd0603r, pcd0604r, pcd0605r, pcd0649r, pcd0650r, pcd0651r, pcd0652r, pcd0653r, pcd0654r, pcd0655r, pcd0656r, pcd0657r, pcd0658r, pcd0659r, pcd0660r, pcd0661r, pcd0662r, pcd0663r, pcd0664r, pcd0665r, pcd0666r, pcd0667r, pcd0668r, pcd0743r, pcd0744r, pcd0745r, pcd0746r, pcd0747r, pcd0748r, pcd0749r, pcd0750r, pcd0751r, pcd0752r, pcd0753r, pcd0754r, pcd0898r, pcd0899r, pcd0900r, pcd0901r, pcd0902r, pcd0903r, pcd0904r, pcd0905r, pcd0906r, pcd0907r, pcd0908r, pcd0909r, pcd0910r, pcd0911r, pcd0912r, pcd0913r, pcd0914r, pcd0915r, pcd0916r, pcd0917r, pcd0918r, pcd0919r, pcd0920r, pcd0921r, pcd0922r, pcd0923r, pcd0924r, pcd0925r, pcd0926r, pcd0927r, pcd0928r, pcd0929r, pcd0930r, pcd0931r, pcd0932r, pcd0933r
cooking utensil	pcd0399r, pcd0400r, pcd0401r, pcd0402r, pcd0403r, pcd0404r, pcd0405r, pcd0406r, pcd0407r, pcd0408r, pcd0409r, pcd0410r, pcd0411r, pcd0412r, pcd0413r, pcd0414r, pcd0415r, pcd0416r, pcd0417r, pcd0418r, pcd0419r, pcd0420r, pcd0421r, pcd0422r, pcd0423r, pcd0424r, pcd0425r, pcd0426r, pcd0427r, pcd0428r, pcd0429r, pcd0430r, pcd0431r, pcd0432r, pcd0433r, pcd0434r, pcd0435r, pcd0436r, pcd0437r, pcd0438r, pcd0485r, pcd0486r, pcd0487r, pcd0488r, pcd0715r, pcd0716r, pcd0717r, pcd0718r, pcd0719r, pcd0720r, pcd0721r, pcd0722r, pcd0759r, pcd0760r, pcd0761r, pcd0762r, pcd1033r, pcd1034r

Label	Images
crockery	pcd0312r, pcd0313r, pcd0314r, pcd0315r, pcd0316r, pcd0317r, pcd0318r, pcd0319r, pcd0320r, pcd0321r, pcd0322r, pcd0323r, pcd0324r, pcd0325r, pcd0326r, pcd0347r, pcd0348r, pcd0349r, pcd0350r, pcd0389r, pcd0390r, pcd0391r, pcd0392r, pcd0525r, pcd0526r, pcd0527r, pcd0528r, pcd0669r, pcd0670r, pcd0671r, pcd0672r, pcd0673r, pcd0674r, pcd0675r, pcd0676r, pcd0677r, pcd0801r, pcd0802r, pcd0803r
electronics	pcd0100r, pcd0101r, pcd0122r, pcd0123r, pcd0124r, pcd0125r, pcd0197r, pcd0198r, pcd0199r, pcd0200r, pcd0256r, pcd0257r, pcd0258r, pcd0259r, pcd0296r, pcd0297r, pcd0298r, pcd0299r, pcd0300r, pcd0301r, pcd0302r, pcd0303r, pcd0304r, pcd0305r, pcd0306r, pcd0307r, pcd0308r, pcd0309r, pcd0310r, pcd0311r, pcd0327r, pcd0328r, pcd0329r, pcd0330r, pcd0351r, pcd0352r, pcd0353r, pcd0354r, pcd0375r, pcd0376r, pcd0377r, pcd0378r, pcd0379r, pcd0380r, pcd0381r, pcd0382r, pcd0383r, pcd0384r, pcd0385r, pcd0386r, pcd0443r, pcd0444r, pcd0445r, pcd0446r, pcd0475r, pcd0476r, pcd0477r, pcd0478r, pcd0489r, pcd0490r, pcd0491r, pcd0492r, pcd0493r, pcd0494r, pcd0495r, pcd0496r, pcd0501r, pcd0502r, pcd0503r, pcd0504r, pcd0517r, pcd0518r, pcd0519r, pcd0520r, pcd0542r, pcd0543r, pcd0544r, pcd0545r, pcd0723r, pcd0724r, pcd0725r, pcd0726r, pcd0797r, pcd0798r, pcd0799r, pcd0800r, pcd1031r, pcd1032r
food	pcd0335r, pcd0336r, pcd0337r, pcd0338r, pcd0691r, pcd0692r, pcd0693r, pcd0694r, pcd0695r, pcd0696r, pcd0697r, pcd0698r, pcd0699r, pcd0700r, pcd0701r, pcd0702r, pcd0703r, pcd0704r, pcd0705r, pcd0706r, pcd0707r, pcd0708r, pcd0709r, pcd0710r, pcd0711r, pcd0712r, pcd0713r, pcd0714r, pcd0804r, pcd0805r, pcd0806r, pcd0807r, pcd0808r,

Label	Images
food	pcd0809r, pcd0810r, pcd0811r, pcd0812r, pcd0813r, pcd0814r, pcd0815r, pcd0816r, pcd0817r, pcd0818r, pcd0819r, pcd0820r, pcd0821r, pcd0822r, pcd0823r, pcd0824r, pcd0825r, pcd0826r, pcd0827r, pcd0828r, pcd0829r, pcd0830r, pcd0831r, pcd0832r, pcd0833r, pcd0834r, pcd0835r, pcd0836r, pcd0837r, pcd0838r, pcd0839r, pcd0840r, pcd0841r, pcd0842r, pcd0843r, pcd0844r, pcd0845r, pcd0846r, pcd0847r, pcd0848r, pcd0849r, pcd0850r, pcd0851r, pcd0852r, pcd0853r, pcd0854r, pcd0855r, pcd0856r, pcd0857r, pcd0858r, pcd0859r, pcd0860r, pcd0861r, pcd0862r, pcd0863r, pcd0864r, pcd0865r, pcd0866r, pcd0867r, pcd0868r, pcd0869r, pcd0870r, pcd0871r, pcd0872r, pcd0873r, pcd0874r, pcd0875r, pcd1001r, pcd1002r, pcd1003r
toy	pcd0247r, pcd0387r, pcd0388r, pcd0479r, pcd0480r, pcd0795r, pcd0796r
work tool	pcd0102r, pcd0103r, pcd0104r, pcd0105r, pcd0118r, pcd0119r, pcd0120r, pcd0121r, pcd0126r, pcd0127r, pcd0128r, pcd0129r, pcd0130r, pcd0131r, pcd0132r, pcd0157r, pcd0158r, pcd0159r, pcd0160r, pcd0165r, pcd0166r, pcd0167r, pcd0168r, pcd0193r, pcd0194r, pcd0195r, pcd0196r, pcd0248r, pcd0249r, pcd0250r, pcd0251r, pcd0284r, pcd0285r, pcd0286r, pcd0287r, pcd0331r, pcd0332r, pcd0333r, pcd0334r, pcd0393r, pcd0394r, pcd0395r, pcd0396r, pcd0397r, pcd0398r, pcd0439r, pcd0440r, pcd0441r, pcd0442r, pcd0505r, pcd0506r, pcd0507r, pcd0508r, pcd0509r, pcd0510r, pcd0511r, pcd0512r, pcd0513r, pcd0514r, pcd0515r, pcd0516r, pcd0521r, pcd0522r, pcd0523r, pcd0524r, pcd0533r, pcd0534r, pcd0535r, pcd0536r, pcd0537r, pcd0538r, pcd0539r, pcd0540r, pcd0541r, pcd0546r, pcd0547r, pcd0548r, pcd0549r, pcd0550r, pcd0551r, pcd0552r, pcd0553r, pcd0554r, pcd0555r, pcd0556r, pcd0557r, pcd0558r, pcd0559r, pcd0560r, pcd0561r, pcd0578r, pcd0579r, pcd0580r, pcd0581r, pcd0582r, pcd0583r,

Label	Images
work tool	pcd0584r, pcd0585r, pcd0586r, pcd0587r, pcd0588r, pcd0589r, pcd0590r, pcd0591r, pcd0592r, pcd0593r, pcd0622r, pcd0623r, pcd0624r, pcd0625r, pcd0626r, pcd0627r, pcd0628r, pcd0629r, pcd0630r, pcd0631r, pcd0632r, pcd0678r, pcd0679r, pcd0680r, pcd0681r, pcd0682r, pcd0683r, pcd0684r, pcd0685r, pcd0686r, pcd0687r, pcd0688r, pcd0689r, pcd0690r, pcd0731r, pcd0732r, pcd0733r, pcd0734r, pcd0735r, pcd0736r, pcd0737r, pcd0738r, pcd0739r, pcd0740r, pcd0741r, pcd0742r, pcd0755r, pcd0756r, pcd0757r, pcd0758r, pcd0787r, pcd0788r, pcd0789r, pcd0790r, pcd0791r, pcd0792r, pcd0793r, pcd0794r, pcd0934r, pcd0935r, pcd0936r, pcd0937r, pcd0938r, pcd0939r, pcd0940r, pcd0941r, pcd0942r, pcd0943r, pcd0944r, pcd0945r, pcd0946r, pcd0947r, pcd0948r, pcd0949r, pcd1000r, pcd1004r, pcd1005r, pcd1006r, pcd1007r, pcd1008r, pcd1009r, pcd1010r, pcd1011r, pcd1012r, pcd1013r, pcd1014r, pcd1015r, pcd1016r, pcd1017r, pcd1018r, pcd1019r, pcd1020r, pcd1021r, pcd1022r, pcd1023r, pcd1024r, pcd1025r, pcd1026r, pcd1027r, pcd1028r, pcd1029r, pcd1030r

APPENDIX B

Calibration and Arm Setup

This appendix details the calibration for the WidowX robot arm (as pictured in Fig. B.1) used throughout these experiments.

Due to the 3-D nature of the setup, both rotation $\mathcal{R}$ and transformation t information is needed, so a rigid transform is estimated using the following method to transform between a grasp in the camera's frame of reference g_i, to a grasp in the robot's frame of reference g_r. This process involves three steps:

1. Finding the centroids between corresponding points in each frame of reference;
2. Bringing both points to an origin and find an optimal rotation $\mathcal{R}$;
3. Find translation t between the frame of reference.

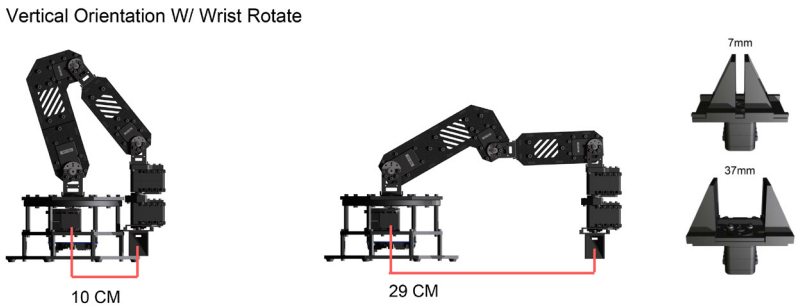


Figure B.1: Diagram of the limits and reach of the WidowX robot arm from www.trossenrobotics.com/widowxrobotarm.

(a) Checkerboard (b) Crosshair (c) Arm placement

Figure B.2: Demonstration of the calibration process for the robotic arm. (a) a checkerboard is placed under the camera, (b) a crosshair is placed at an intersection, (c) the centre of the end-effector is placed at the same intersection. This is repeated around 10 times at different points and a rigid transform is calculated.

Firstly, the centroid is the average of corresponding points in the camera frame and robot frame of reference. To do this, the process is followed as in Fig. B.2, a visible checkerboard pattern is placed in the centre of the field of view of the camera (Fig. B.2a). The intersection between points on the checkerboard is used for consistency. A point is chosen on the board (Fig. B.2b), and the centre of the robot end-effector is manoeuvred to match the given point on the board (Fig. B.2c). This process is repeated about 10 times at different points on the board (however, only a minimum of three points are needed). The average points are calculated to produce a centroid $\mathcal{C}$ in the camera frame of reference $\mathcal{C}_\rangle$ and the robot frame of reference $\mathcal{C}_\nabla$:

$$\mathcal{C}_i = \frac{1}{N}\sum_{i=1}^{N} A, \mathcal{C}_r = \frac{1}{N}\sum_{i=1}^{N} B, \tag{B.0.1}$$

where A and B are the sets of $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ vectors gathered during the calibration process. For the points from the camera A, x and y are the pixel coordinates taken from the camera, and z is the raw depth value taken from the depth camera. For B, these values are saved directly from the robot, and works for any coordinate system as the rotation element works it out automatically.

The optimal rotation R is calculated using singular value decomposition (SVD) from the built in *numpy* library in *Python*, to decompose a covariance matrix H into the matrix

$\left[U, S, V\right]$ and solving for the optimal rotation R:

$$H = (A - \mathcal{C}_i)(B - \mathcal{C}_r)\left[U, S, V\right] = SVD(H)\mathcal{R} = VU^T, \tag{B.0.2}$$

where H is a 3×3 matrix calculated through the operation $A - \mathcal{C}_i$, which subtracts each column in A by centroid $\mathcal{C}_i$ and the same for $B - \mathcal{C}_r$. This represents bring each centroid to the origin to more easily find the rotation between the two sets of points. A special reflection case is added by checking whether the determinant of $\mathcal{R}$ is negative, as sometimes SVD returns a reflection matrix. In this case, the third column of V is multiplied by -1.

The transformation t between the two frames can then be solved by:

$$t = \mathcal{C}_r - \mathcal{R} \times \mathcal{C}_i, \tag{B.0.3}$$

to produce a rotation and transformation between the two frames.