Institut für
Mechatronische Systeme
imes Leibniz Universität Hannover

Leibniz
Universität
Hannover

A Deep Learning-Based Robotic Grasping Framework with RGB-D Sensors

Masterarbeit M-03/2020-932

Lei Zhang
Matrikelnummer 10014875

Hannover, 09. Mar. 2020

Erstprüfer	Prof. Dr.-Ing. Tobias Ortmaier
Zweitprüfer	Prof. Dr.-Ing. Jörg Wallaschek
Betreuer	Max-Heinrich Laves, M.Sc.
Betreuer	Jun Deng, Dr.-Ing.

Ihr Ansprechpartner:
Dr.-Ing. Jun Deng

+ 49 (0)8105 3980-106
jun.deng@agile-robots.com

2.10.20

Gewährung von Nutzungsrechten

Hiermit gewähre ich im Rahmen meines Urheberrechts an meiner Masterarbeit zum Thema *„A Deep Learning-based Robotic Grasping Framework with RGB-D Sensors“* zeitlich unbeschränkt Nutzungsrechte in vollem Umfang sowohl an Herrn Dr.-Ing. Jun Deng, als auch an Agile Robots AG.

Die gewährten Nutzungsrechte schließen dabei ausdrücklich jedwede Veröffentlichung der Ergebnisse und Darstellungen sowie eine nichtkommerzielle und kommerzielle Nutzung der Ergebnisse und Darstellungen ein.

10.02.2020

Ich, Lei Zhang, versichere hiermit, dass die vorliegende Arbeit selbstständig verfasst wurde, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden, alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen hat.

Hannover, 09. Mar. 2020

(Lei Zhang)

Institut für Mechatronische Systeme
Leibniz Universität Hannover

Leibniz Universität Hannover

Aufgabenstellung zur Studienarbeit

Lei Zhang, Matrikelnummer 10014875

Allgemeines:
Bildgestützte Automatisierung, maschinelles Lernen und künstliche Intelligenz sind aktuell die Forschungsschwerpunkte, wenn es um die Technologie der Zukunft geht. Ein Beispiel aus der Robotik stellt das Bin-Picking, auch „Griff in die Kiste“ genannt, dar. Objekte mit verschiedenen Formen, Oberflächen und unterschiedlicher Nachgiebigkeit sind zufällig in einer Kiste angeordnet. Mit Hilfe von Bildbearbeitungsalgorithmen, basierend auf künstlicher Intelligenz, sollen die Objekte erkannt und durch einen Roboter gegriffen werden. Sowohl in der Wissenschaft als auch in der Industrie wurden bereits Lösungen umgesetzt.

Aufgabe:
Im Rahmen dieser Masterarbeit soll eine effektive Lösung der Robotic-Grasping-Aufgabe erarbeitet werden. Bei dem System wird eine Robotic-Grasping-Lösung unter Verwendung von Datagenerierung und End-to-End-Learning entwickelt. Eine Datenbank für Robotic-Grasping durch physikallische Model wird erstellt, damit wird ein Deep Learning basierend auf künstliche neuronale Netze aufgebaut, um das Greifen zu prognostizieren. Weiterführend soll aufgrund der hohen zu erwartenden Rechenleistung für Künstliche-Intelligenz-Algorithmen, insbesondere neuronale Netze, eine Untersuchung zur Demonstrationsanwendung des Systems durchgeführt werden. Besondere Beachtung wird dabei der Stabilität der Lösung und deren Evaluierung gewidmet.
Im Rahmen dieser Arbeit ergeben sich insbesondere die folgenden Aufgabenpunkte:

- Literaturrecherche zum Robotic-Grasping (Algorithmen, Systeme)
- Entwicklung und Aufbau der Robotic-Grasping-Datenbank
- Entwicklung und Aufbau des Neuronalen Netzwerk-Framework
- Entwicklung und Aufbau eines geeigneten Robotic-Grasping-Systems
- Evaluierung des Systems
- Entwicklung einer Demonstrationsanwendung

Die Bearbeitungszeit beträgt 900 Stunden.

Ausgabe der Aufgabenstellung:	09.09.2019
Abgabe der Arbeit spätestens am:	09.03.2020
Prüfer:	Prof. Dr.-Ing. Tobias Ortmaier
Zweitprüfer:	Prof. Dr.-Ing. Jörg Wallaschek
Betreuer:	Jun Deng, Dr.-Ing.
Betreuer:	Max-Heinrich Laves, M. Sc.

Abstract

Grasping objects is a fundamental property of robotics and recently data-driven approaches have advanced the science of robotic grasping. In addition to the traditional computer vision techniques, the methods emerging through deep learning can also be utilized for robotic grasping. To reliably complete effective robotic grasping based upon deep learning and economize upon the costs of the data collection required for such a task, this paper attempts to build a deep learning-based robotic grasping framework with RGB-D sensors consisting of the whole pipeline from data generation to model inference.

For the robotic grasping framework, main tasks were defined: the data generation required for reliable grasping using analytical methods; a deep learning-based framework for robotic grasping detection, in which lower precision in terms of sensing and control were considered; and the evaluation on a real-world platform.

Accordingly, this study aimed to generate the datasets for unknown object grasping tasks to build a robust and powerful deep learning-based robotic grasping framework with RGB-D sensors that allowed the images to be fed in and the grasping pose extracted. Through our data generation method, the grasping sample was extracted taking into account the geometric information of the 3D model to avoid a label imbalance problem. In addition, the grasp quality was estimated using an analytical method, that is, the force-closure and Ferrari-Canny metric, and we presented a Gaussian distribution-based data augmentation method for robotic grasping data synthesis. Finally, we generated several robotic grasping datasets with analytical and statistical methods, which displayed a high adaptability to the camera in experimental setups. Our network was trained using synthetic datasets and we utilized the global optimization method, in view of the force-closure metric and gripper model, to select the executed grasp from candidates in the grasp affordance map for robotic grasping on the real-world robotic experiment platform.

In our physical robotic experiments, our grasping framework had success rates of 90.91% for grasping single known objects and novel household objects with complicated shapes and 85.71% for multi-objects robotic grasping in a complex scenario. The experimental results confirm that our data-driven deep learning-based robotic grasping framework achieves the state of the art for robust and efficient robotic grasping with a low-resolution 3D camera.

Keywords: Deep Learning, Robotic Grasping, Data Generation, Transfer Learning

Contents

List of Figures

List of Tables

Acronyms

3D	3 Dimensional
6D	6 Dimensional
CNN	Convolutional Neural Network
OWS	Object Wrench Space
GWS	Grasp Wrench Space
Dex-Net	Dexterity Network
GQ-CNN	Grasp Quality Convolutional Neural Network
DOF	degree of freedom
MLP	multilayer perceptron
ReLU	Rectified Linear Unit
tanh	hyperbolic tangent Activation Function
HDF5	Hierarchical Data Format
RGB	Red, Green, and Blue
DLR	German Aerospace Center (Deutschland für Luft- und Raumfahrt)
GMM	Gaussian Mixture Model
Syn-Train	Synthetic Training Set
YCB	Yale-CMU-Berkeley dataset for robotic manipluation research
GO	global optimization
VGG	Visual Geometry Group at University of Oxford
AlexNet	A CNN, designed by Alex Krizhevsky
GoogLeNet	Networks with Parallel Concatenations
ResNet	Residual Network
COCO	Common Objects in Context

1 Introduction

In robotics and manipulation, one of the most important controlled tasks in terms of perception is grasping. As the old saying goes, "Man is the most intelligent of the animals because he has hands" [Bic00]. In addition to the guided regulation of hand manipulation in the design of robot hands, the main criterion of success is attaining a grasping pose for a defined target. In the case of robotic grasping, researchers focus on the mechanism design, motion planning, and grasp planning, which include grasp detection, visual servo control, that is, vision-based robot control, and grasp-point optimization [Car12]. Traditionally, grasp detection has depended upon the human knowledge to empirically generate the grasp pose according to the task. However, with the rapid development of artificial intelligence in the last years, deep learning-based methods for robotic grasping have been extensively applied to improve the overall performance of this task [CRC18] [Car12]. In this thesis, a new deep learning-based robotic grasping framework based on RGB-D sensors is presented that utilizes the aspiring methods of data generation and deep learning. In recent times, the process of deep learning has thrived as a result of big data and has demonstrated great potential in a broad area of applications such as computer vision, speech recognition, and natural language processing. Nonetheless, data collection has been a significant challenge in the field of deep learning, while the preparation of the data is time-consuming, involving collection, pre-processing and feature engineering. For this reason, data generation is more frequently being applied to collecting the data required for deep learning due to its low costs and flexibility [RHW19].

1.1 Problem Statement

Numerous interactive and collaborative robots, as a kind of benign collaboration between human beings and robots, are increasingly being used for pick-and-place activities, packaging, and palletizing within the automotive logistics factories and advanced manufacturing. Given that grasping is one of the most important skills of collaborative robots, it plays an essential role in these tasks [KGK+18].
A deep learning-based robotic grasping framework with a three-dimensional (3D) camera has long been employed for tasks from object localization, pose estimation, and grasp detection to motion planning [DWL19]. However, it is recognized that high-precision empirical methods aimed at performing grasping tasks involving unknown objects (model-free grasping) remain a challenge due to the limited number of objects that currently exist in the grasp database. Additionally, the current large-scale database for robotic grasping detection is extraordinarily limited. Nevertheless, a robust and efficient network with high generalization training from a small amount of data is not a straightforward training task. At the same time, the geometric and physical similarities between the 3D objects to be grasped and those of the annotated models are extremely critical to the outcome of robotic grasping, which gives rise to the problems in data generation. In this regard, the human annotation method for generating databases has

a bad reputation for its low speed and unstable results with regard to the labels it produces [DWL19]. Hence, data generation for robotic grasping is critical for a robust robotic grasping framework.
This thesis is specifically concerned with data generation and deep learning methods utilized in conjunction with a 3D camera. It aims to yield a deep learning-based framework for effectual and robust model-free robotic grasping and build a database with large numbers of grasping samples for vision-based robotic grasping using analytical grasping quality metrics. To this end, the goal of the thesis is divided into the following aspects:

- The development of data generation intended for robotic grasping.
- The generation of grasping datasets with analytical grasp measure methods.
- The construction of a pipeline for the robust robotic grasping framework.
- The evaluation of the robotic grasping framework.
- Comparative experiments of the framework within a real-world robotic grasping platform.

1.2 Significance of the Study

Due to the lack of an image-based grasping database as well as the expanding applications of robotic grasping in automation, this thesis endeavours to present a robust and productive deep learning-based framework with RGB-D sensors for model-free robotic grasping tasks, which combines a data generation approach and data-driven deep learning methods. In particular, we aim to generate an image-based dataset with a large number of samples and reliable labels based upon analytical methods, which play a critical role for the data-driven deep learning approach. At present, robotic grasping with a low-resolution camera is still a challenging topic in the field of research and industry. Consequently, the robotic grasping methods based on a low-resolution camera are also considered within this thesis.

1.3 Thesis Structure

This thesis is divided into the following six chapters. Chapter 1 introduces the background of robotic grasping, data synthesis, and deep learning. The statement of the problem, the significance of the study and its contents are specially described. In Chapter 2, the state of art concerning the use of deep learning-based methods for robotic grasping is presented. Chapter 3 contains the conceptual and theoretical framework of the study. The separate components of our framework are described in detail, including the basic concepts in contact model and the convolutional neural network (CNN). In Chapter 4, the methods used in the work are presented, including the data generation methods, the data preprocessing, the network architecture, the training process of the CNNs, and our grasp execution pipeline. Chapter 5 provides the experimental framework, including the robotic and camera configurations, the comparative experiments using different data generation implementations and various parameters during the training process and execution as well as the results of the experiments. The functionality of the

deep learning-based robotic grasp framework is also investigated in the qualitative study. Finally, the conclusions formed regarding the novel features introduced and a discussion of the improvements and future research possibilities are presented in Chapter 6.

2 The State of the Art

As a result of the advancements in optical sensors over recent decades, vision-based frameworks are now widely used in the field of robotic grasping. Traditionally, vision-based robotic grasping frameworks consisted of different components, including object localization, object pose estimation, grasp detection, and grasp planning [DWL19][Car12]. Object localization and pose estimation, respectively, provide the position and orientation information of the objects to be grasped for the robotic grasp framework. Following this, the grasping methods are established according to the grasp detection algorithm which calculates the grasp configuration with regard to the object. Lastly, the grasp planning, including the picking and placing of an object, can be implemented and the whole grasping process is completed [DWL19].

2.1 Robotic Grasping Methods

Robotic grasping methods are classified according to grasp detection in the following ways: analytical approaches, empirical approaches, data-driven methods, deep learning-based methods, and others [DWL19].
The analytical methods refer to the grasp detection based on the geometric, kinematic and/or dynamic formulations required to achieve the stability, force-closure, and task compatibility designed for grasping. The force closure means that the fingertips are able to grasp object so as to resist any external force or moment [SEKB12]. Overall, the analytical approaches can be divided into three groups: force-closure grasp synthesis for 3D objects, optimal force-closure grasp synthesis with a grasp quality metric, and task-oriented methods. It is generally known that force closure is the basic precondition of grasping. At the same time, the force-closure condition is converted into the calculation of the angle between the normal directions on the object's surface or an analytical formulation using a linear model, in which the contact facets of grasping are not considered [Ngu87] [DLSX00]. There are also some methods that consider the force-closure and the grasp quality criteria based on the object's geometric information. In this regard, to evaluate grasp quality, the concept of object wrench space (OWS) was introduced by Pollard, which provides the information about the object geometry. Specifically, OWS is the set of wrenches that can be produced through the external force acting on one contact point on the object surface in which the torque values in the wrench are scaled according to the longest axis of the object for the purpose of maintaining the scale invariance [BFH04] [SEKB12]. Following this, the grasp wrench space (GWS) was established in order to quantify the quality of the grasp, which incorporates all the wrenches that are created by the forces on the grasping contact points with force-closure [BFH04] [MKCA03] [SEKB12]. To this end, Ferrari and Canny introduced a method to approximate the calculation process of GWS. This is known as the Ferrari-Canny grasp quality metric or the Ferrari-Canny metric, which

computes grasp qualities for force-closure grasps for the planning of optimal grasps [CF92] [BFH04]. However, compared to the expensive computation involved in the analytical methods, the empirical methods refer to the learning and classification approaches that observe the human operators or the relationships between the object's properties to complete the task [SEKB12]. For example, Zhou from Google Brain (a deep learning artificial intelligence research team at Google) presented a method called "Watch, Try, Learn", in which agents were able to learn to learn from demonstrations and trial-and-error to accomplish grasping, pick-and-place, pushing, and button pressing tasks [ZJK+19].
Subsequently, the data-driven methods have been widely used to transfer robotic grasping based upon the mass of 3D models to the real-world object. These approaches are divided into grasping methods based on whether they are known or unknown objects, and, in turn, the grasping methods based upon the known and unknown objects are respectively referred to as model-based and model-free methods [BMAK13]. In the model-based robotic grasping process, the grasped object is detected by the object recognition method and the pose estimation approach. One major drawback of this approach is that the methods are sensitive to sensor noise and imprecise data. In particular, pose estimation methods based upon color or depth images react to the sensor quality and environmental factors, which may impact the input data. As a result of the negative factors of model-based robotic grasping, developers have deliberated upon building a robust model-free method for robotic grasping. In the model-free grasping methods, the particular features are extracted that are able to quantify the success rate of robotic grasping analytically or empirically [BMAK13][DWL19].
The final method used for grasp detection is that of deep learning, a subset of machine learning in artifical intelligence (AI), which is able to learning from data. The deep learning-based methods for robotic grasping without 6D pose estimation are rapidly increasing, including supervised learning methods, reinforcement methods, and others [DWL19]. The deep learning-based method without pose estimation means that the result of grasp detection can only be accomplished with input data. In this regard, Mahler et al. have presented the robust robotic grasping framework Dex-Net 2.0 with Grasp Quality Convolutional Neural Network (GQ-CNN) to predict the possibility of grasping success with parallel-jaw grippers. To do so, they generated a database containing more than 500,000 images with the local contact features of grasping. Dex-Net 2.0 performs highly efficiently in robotic grasping tasks, in spite of lacking the global features of the objects and containing a large number of parameters, which reduce the flexibility of the framework [MLN+17]. Recently, the Dex-Net MM was applied to the robust grasping of a mobile robot and achieved good content performance [STM+19] [DWL19]. In addition, end-to-end learning methods have been broadly utilized in grasp detection. Many researchers have employed the deep learning method to predict graspable bounding boxes with the grasping orientation and position using the sensor image as input [RA15] [LLS15] [ZZL+19]. Moreover, pixel-wise deep learning methods have been employed to predict the grasp quality, gripper width, and grasp direction for every pixel in the input image[MCL18][ATH19]. There are also the deep learning-based methods PointNet and PointNet++, which are able to extract the features containing the position and orientation information from 3D data [QSMG17] [QYSG17]. Liang et al. utilized PointNet for grasp detection and Mousavian et al. from NVIDIA presented a 6-DOF GraspNet for predicting grasp by observing the point

clouds from a depth camera [LML+19] [MEF19]. However, the uncertainty of the deep learning-based methods remains high due to database and other related factors.

3 Foundations

In this thesis, we present a grasp detection framework based on data generation and the deep learning-based method to efficiently and robustly predict the grasping pose. In this chapter, the foundations of the data generation based upon the analytical methods and the CNN are introduced individually. Primarily, we describe the basic concepts of the contact model: force-closure method and grasp quality metric, which are generally utilized in grasping with a two-jaw-gripper. The force-closure grasp can be estimated based on friction cone and wrench space method and the force closure means that the fingertips of gripper could resist any external forces and moments executed on the contact points. To quantify the grasp quality of the force closure grasps, the grasp quality metric of Ferrari-Canny metric is utilized in force closure grasp sampling during the data generation process. In addition, fundamental knowledge about the deep learning-based methods used in our thesis is incrementally identified.

3.1 Basic Concepts of the Contact Model

In this section, the basic concepts of the contact model: force-closure and the grasp quality metric, are presented in detail. Firstly, it should be noted that the force-closure grasp means that the motion of the grasped object is absolutely constricted, and force-closure is a basic requirement for a stable grasp. With the force-closure method, we can estimate the possible contact points on the surface of the 3D model at the outset. Additionally, the optimum force-closure grasp can be evaluated analytically with grasp quality metrics, which consider the contact model and wrench space. These analytical methods were effectively applied in our pipeline for generating robust grasping with a two-jaw gripper.

3.1.1 Coulomb Friction Model and Friction Cone

Coulomb friction is a physical model used for analyzing the frictional forces in robotic manipulation. Coulomb friction states that the tangential friction force $\boldsymbol{f}_t$ is associated with the normal force magnitude $\mu \boldsymbol{f}_n$ and the formula can be simplified as follows:

$$\boldsymbol{f}_t \leq \mu \boldsymbol{f}_n \tag{3.1}$$

The μ represents the friction coefficient and, for the sake of simplicity, the simplest Coulomb friction model with a constant friction coefficient was used in this thesis. The friction coefficient depends upon the material properties of the gripper fingertips and objects. Normally, the friction coefficient is in the range of 0.1 to 1 [LP17].
Typically, the forces that can be applied through the contact points meet the following condition and form a friction cone at the contact when the normal of contact point is in the $\widehat{z}$ direction.

$$\sqrt{\boldsymbol{f}_x^2 + \boldsymbol{f}_y^2} \leq \mu \boldsymbol{f}_z \tag{3.2}$$

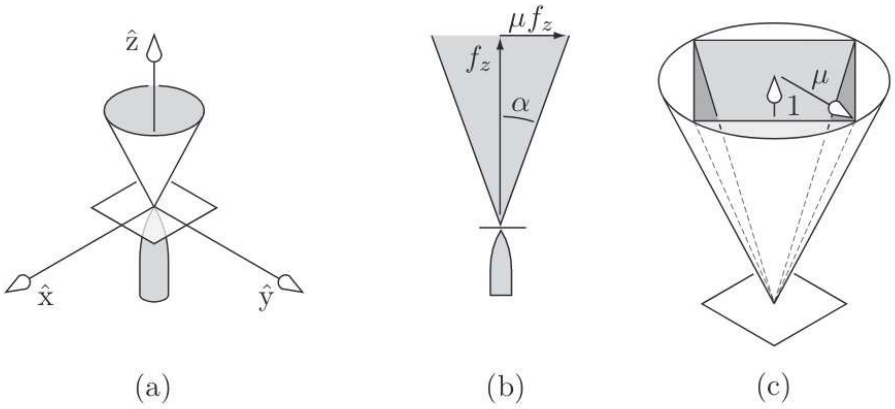


Figure 3.1: Illustration of a friction cone. (a) A friction cone with all the possible forces that can be exerted through contact. (b) The side view of a friction cone (μ: friction coefficient, $\alpha = \tan^{-1}\mu$). (c) The polyhedral convex cone approximation [LP17].

To approximately calculate the contact mechanics, the polyhedral convex cone in Figure 3.1 (c) represents the friction convex cone. Figure 3.1 (c) demonstrates that the friction cone can be mathematically approximated with the four-sided pyramidal cone for computational purposes [LP17]. As the numbers of pyramid faces increases, the errors between the pyramidal convex cone and the convex circular cone decrease accordingly and the parameters of computation increase accordingly [Ngu87].
Typically, the fingertip contacts can be divided into two types: hard-finger and soft-finger contacts. The hard-finger contact can be described by means of the friction cone without torques in the contact and the soft-finger contact can be simulated with the forces pointing into the friction cone and the corresponding torques [Ngu87].
Taking into consideration the number of computational parameters and the accuracy of the analytical results, the octagonal friction cone was utilized during the data generation required for robotic grasping.

3.1.2 Wrench Space

In order to describe the force and corresponding moment on the contact point, the wrench is defined with the force and the torque acting at a point as the following formulation [Mur17]:

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix} \quad \begin{array}{ll} \boldsymbol{f} \in \mathbb{R}^3 & \text{force at the contact point.} \\ \boldsymbol{\tau} \in \mathbb{R}^3 & \text{torque at the contact point.} \end{array} \tag{3.3}$$

The force/torque pair refers to a wrench and the torque is the product of the force and the distance to the object from the reference point. During our method, the reference point was the center of a 3D model. The wrench space announces the set of all wrenches that can be transmitted to the object through the contacts and wrench space has been usually utilized in evaluating the friction model.
As contact friction can create torques, a 6-dimensional wrench cone $\boldsymbol{F} = ([p]\boldsymbol{f}, \boldsymbol{f})$ is correspondingly defined as the representation of the friction cone, where $\boldsymbol{p}$ and $[p]\boldsymbol{f}$ separately represent the location of the contact and the moment created by the friction [LP17]. As shown in Figure 3.2, the wrench cones can be plotted with the friction forces and torques in the coordinate system $(\boldsymbol{f}_x, \boldsymbol{f}_y, \boldsymbol{m}_z)$. In Figure 3.2 (a) and (b) illustrate that the wrench cone with a single contact. When two contacts are considered, the wrenches can be transferred to the object through the contact points, which are the sum of the positive span of all the single wrench cones WC_i [LP17]:

$$\boldsymbol{WC} = pos(\boldsymbol{WC}_i) = \{\sum_i k_i \boldsymbol{F}_i \mid \boldsymbol{F}_i \in \boldsymbol{WC}_i, k_i \geq 0\} \tag{3.4}$$

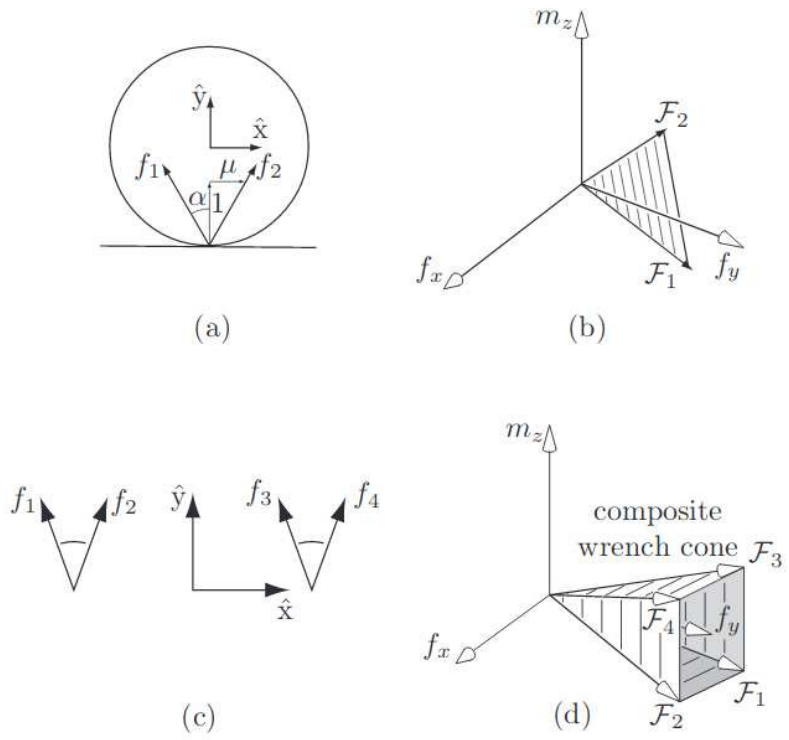


Figure 3.2: Illustration of wrench space. (a) A planar friction cone (μ: friction coefficient, $\alpha = \tan^{-1}\mu$). (b) The corresponding wrench cone. (c) Two friction cones with multiple contacts on the object. (d) The corresponding wrench space with two contacts. [LP17]

Specifically, with regard to grasping with n contact points, the unit GWS can be defined as the set of unit wrench spaces at each contact point [LS16]:

$$\boldsymbol{WG} = \{\boldsymbol{w} \mid \boldsymbol{w} = \sum_{i=1}^{mn} k_i \boldsymbol{F}_i, k_i > 0, \sum_{i=1}^{mn} k_1 = 1, \mid \boldsymbol{w}_i \mid = 1\} \tag{3.5}$$

Namely, the unit GWS can be described as the set of all the possible wrenches that can be applied by all the contacts if applying unit magnitude of contact force.

3.1.3 Force-Closure Grasp

The force-closure grasps establish the separate space of contact for the fingertips of the gripper, which ensures that any external wrench on the object can be balanced by the contact forces. Consequently, we can define that a grasp as a force-closure grasp if given any external wrench $\boldsymbol{F}_e$ is executed upon the object, the contact forces $\boldsymbol{f}_c$ meet the condition [Mur17]:

$$G\boldsymbol{f}_c = -\boldsymbol{F}_e \tag{3.6}$$

This also means that the grasped object is constrained by the fingertips [LP17].
Specifically, the force-closure problem can be divided into two parts: (1) force-direction closure and (2) torque closure [Ngu87]. The force-direction closure is tested by the faces of the contact model, while the torque closure is affected by the placement of the fingertips.
Accordingly, the condition of force closure can be described with the following formulation:

$$\boldsymbol{G}\boldsymbol{f}_{contact} + \boldsymbol{F}_{external} = 0, \boldsymbol{G} = \begin{bmatrix} n_{c_1} & \cdots & n_{c_k} \\ p_{c_1} \times n_{c_1} & \cdots & p_{c_k} \times n_{c_k} \end{bmatrix} \tag{3.7}$$

where n_c refers to the unit normal force in the contact point.
Furthermore, the methods of checking force-closure can be divided into the following steps:
1. Use the linear combinations of the forces that describe the edges of the approximated friction cone to express every contact force.
2. Check whether a grasp is force-closure by researching whether the OWS contains a neighborhood of the origin. If the wrench space contains a neighborhood of the origin, the grasp is force-closure.
According to the work of Nguyen, a grasp with two soft-finger contacts is force-direction closure if and only if the angle ϕ between two contact planes is less than the angle of friction cone 2θ. Then the force-direction closure condition reduces to the simple relationship between the angle between two normal directions in the contacts. When the reverse contact forces are considered, the torque perpendicular to the line segment P_1P_2 pointing to the contact points can calculated. The 3D models of the contact model and friction cones in the contacts are shown in Figure 3.3.
Based on the contact model in Figure 3.3, as previously stated, Nguyen established that the conditions of force-closure, which means that a grasp with two soft-finger contacts is force-closure if and only if the

following two conditions are met:

- $\phi < 2\theta$
- the segment P_1P_2 or P_2P_1, points into and out of the two friction cones at the contact points P_1 and P_2 [Ngu87].

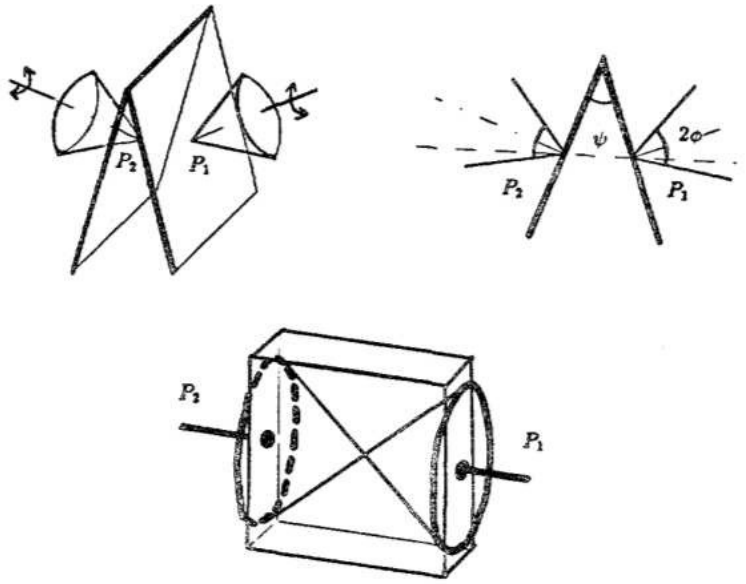


Figure 3.3: Force-closure with two soft-finger contacts. [Ngu87]

Consequently, a simple method of estimating a force-closure grasp, which consists of two contact points, is presented based upon the following theorem [Mur17]:

Theorem 1 (Planar antipodal grasps) *A planar grasp with two point contacts with friction is force-closure if and only if the line connecting the contact point lies inside both friction cones.*

A grasp that meets the conditions of the theorem is an antipodal grasp. During the study in this thesis, the conditions of an antipodal grasp were utilized in sampling the grasps according to the object meshes and friction cones.

3.1.4 Grasp Wrench Space

In Sec. 3.1.2, we have described wrench space. The GWS refers to the set of wrenches that can be executed upon the grasp points of the grasped object.

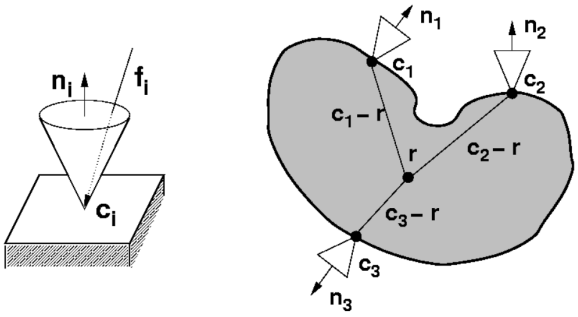


Figure 3.4: Example of GWS. [BFH04] Left: Illustration of the friction cone of a single grasp contact point. Right: A example of the grasp with three contact points.

An illustration of the friction cone of a contact point is shown in left of Fig. 3.4. The force $\boldsymbol{f}_i$ can be applied to the contact point $\boldsymbol{c}_i$ according to Equation 3.2. Each force $\boldsymbol{f}_i$ generates one torque related to the inference point r, as in the following equation [BFH04]:

$$\boldsymbol{\tau}_i = (\boldsymbol{c}_i - \boldsymbol{r}) \times \boldsymbol{f}_i \tag{3.8}$$

Then, the wrench space $\boldsymbol{w}_i$ can be defined with the force $\boldsymbol{f}_i$ and torque $\boldsymbol{\tau}_i$ [BFH04]:

$$\boldsymbol{w}_i = \begin{pmatrix} \boldsymbol{f}_i \\ (\boldsymbol{c}_i - \boldsymbol{r}) \times \boldsymbol{f}_\mathbf{i} \end{pmatrix} \tag{3.9}$$

In order to detail the GWS, the Cone Wrench Space (CWS) is defined as follows [BFH04]:

$$\boldsymbol{CWS}_{\boldsymbol{c}_i} = \left\{ \boldsymbol{w}_i | \boldsymbol{w}_i = \begin{pmatrix} \boldsymbol{f}_i \\ (\boldsymbol{c}_i - \boldsymbol{r}) \times \boldsymbol{f}_i \end{pmatrix} \wedge \| \boldsymbol{f}_i - (\boldsymbol{f}_i \cdot \boldsymbol{n}_i)\, \boldsymbol{n}_i \| \leq -\mu\, (\boldsymbol{f}_i \cdot \boldsymbol{n}_i) \wedge \| \boldsymbol{f}_i \| \leq 1 \right\} \tag{3.10}$$

The CWS consists of a set of wrenches in which the forces meet the Coulomb friction model and the applied forces are smaller than a value of 1.
Then, the GWS is constructed with consideration of all the grasp points. The set of wrenches in the GWS is shown in the following formulation [BFH04]:

$$\boldsymbol{GWS} = \left\{ \boldsymbol{w} | \boldsymbol{w} = \sum_{i=1}^{k} \boldsymbol{w}_i \wedge \boldsymbol{w}_i \in \boldsymbol{CWS}_{\boldsymbol{c}_i} \right\} \tag{3.11}$$

3.1.5 Grasp Quality Metric

As previously stated, the GWS refers to the set of all the wrenches that can be transmitted to the object through the grasp contacts. In order to evaluate the grasp quality, the wrench space with various grippers was used. In addition, different measurement methods were utilized in the calculations of the GWS

[BFH04].
In this regard, the German Aerospace Center (DLR) established a fast and robust grasp planner for arbitrary 3D objects by calculating the convex hull using the discretization method for the friction cones. The experiments based upon grasping with the DLR Hands, one of the most advanced and complex artificial hands in the world, proved that the discretization has a large influence on the errors in the approximation of the GWS. Unequivocally, four vectors will lead to an error of 30% and eight vectors will lead to an error of 8% [BFH99]. Accordingly, Ferrari and Canny proposed an approach to calculate the force-closure property with the magnitude of the contact wrenches that can compensate for the disturbance wrench in the worst case [CF92] [BFH04] [LS16]. Primarily, the force-closure condition was used to obtain suitable grasp samples. The friction cones were estimated first and the wrenches of each force were calculated based upon the estimated friction cones. Following this, the contact model was able to be described with the formulation 3.7. The grasp contact model will be processed according to calculation of wrench space to get the boundary of wrench space BG, which means corresponding conditions in Equation 3.7 . The grasp contact model was processed according to the calculation of the wrench space to obtain the boundary of the wrench space BG, corresponding to the conditions in Equation 3.7. Furthermore, the wrench modules on the surface of the GWS convex hull were considered to be candidates for calculating the grasp quality. Finally, the distance of the candidate was treated as the grasp quality Q, which is nearest to the origin, as in the following formulation:

$$Q = \min_{\boldsymbol{w} \in \text{ConvexHull}(BG)} \|\boldsymbol{w}\| \tag{3.12}$$

It is also known that the Q is the radius of the inscribed sphere in the convex hull of the GWS. Specifically, all the wrenches are scaled until each applied force at a contact point is unit force before the estimation. Hence, the weakest candidate with the lowest distance to the origin holds the minimum wrench [CF92].

3.1.6 Stable Pose on the Planar Table

This subsection describes the algorithm "compute_stable_poses" used for estimating the stable pose of an object on a planar surface from Trimesh library [Daw], which provides a pure Python Library for loading and using triangular meshes.
Using the algorithm, the stable orientation and quasi-static probability of a mesh can be estimated. The algorithm samples the location of the center of mass, based upon a multivariate Gaussian distribution over n_samples. The sampling process is executed multiple times during the estimation. For each sample, the stable resting poses of the mesh on a planar workspace are computed and the probabilities of landing in each pose if the object is dropped onto the table randomly are evaluated. The results of the algorithm contain the 4×4 transform matrices that place the shape against the planar surface with the z-axis pointing up along with the possibilities for every estimated pose. Specifically, the stable pose refers to the estimated pose on a planar surface in the following section of the thesis.

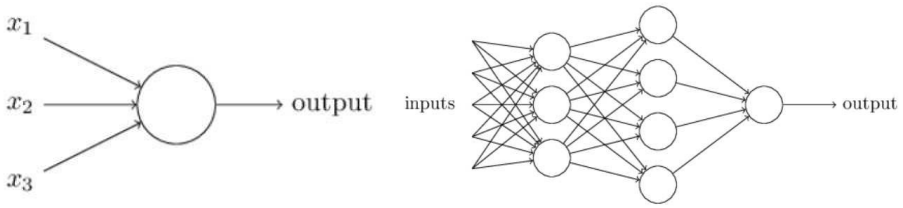
3.2 Basic Concepts in Convolutional Neural Networks

The receptive field was established in the 1960s and in the 1980s convolutional neural network (CNN) was defined by the Neocognitron network invented by Fukushima, which refers to the region in the input space that a CNN's features are affected by. In recent years, CNN has been utilized in finishing different tasks efficiently. A typical CNN contains convolutional layers, pooling layers, and fully connected layers. By adding multilayers and components within one layer, a deep neural network can be created to represent the functions required for supervised learning, which could predict and calculate the result of a task. Most tasks can be mapped to the calculations from an input vector to an output vector and the mapping can be performed by deep learning with a network model and a large dataset with labeled samples [GBC16].
This section describes firstly the feed-forward neural networks that were used to represent the functions of the task. Additionally, the core technologies in the training process will be detailed that explain how the neural network can learn the information from the dataset and the prediction of the output vectors are detailed. Furthermore, the CNNs that are often used in image processing, training processes and the network architectures are briefly introduced. Lastly, we propose the definitions of transfer learning used in our network training procedure.

3.2.1 Neural Networks

Perceptrons

To begin to understand a neural network, perceptrons, a type of artificial neurons, are detailed in this subsection.
When many perceptrons are connected together, a multilayer perceptron (MLP; Fig. 3.5 b) is created. This type of network can be quite powerful. As shown in Figure 3.5 (b), the perceptrons with inputs can be called the input layer. The intermediate layers, called hidden layers, extract features from the input data. The output layer in the last layer predicts the final results [PBPPM09] [Nie15].



(a) (b)

Figure 3.5: Perceptron and multilayer perceptron. [Nie15]

Feed-Forward Neural Networks

The MLPs, also called feed-forward neural networks, are the essential deep learning models. In deep feed-forward neural networks, the function f^* that represents the calculation of multiple layers is approximated. Specifically, the classifier for grasping $y = f^*(x)$ refers to the category y of one grasp pose. At the same time, the parameters w in the function $f(x, w)$ are approximated while the deep feed-forward neural networks learn features from the labeled data [GBC16]. Feed-forward neural networks mean that the information from the input data will flow through the input layer and the hidden layers and lastly to the predicted result y. In addition, there is no feedback information from the output. Hence, the formulation for the network in Figure 3.5 (b) can be defined as the following:

$$f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x))), \tag{3.13}$$

$f^{(1)}$: input layer ,
$f^{(2)}$: hidden layer,
$f^{(3)}$: output layer [GBC16].

Meanwhile, the basic function in the first layer can be constructed in the form: [Bis06]

$$f(\boldsymbol{x}_j, \boldsymbol{w}) = \sum_{i=1}^{M} \boldsymbol{w}_{ji}\boldsymbol{x}_j + \boldsymbol{w}_{j0} \tag{3.14}$$

In order to simulate the function of the neurons, the non-linear activation function is added after the basic functions. Correspondingly, an activation function for Equation 3.1 is used for the classification problems. [Bis06]

$$\boldsymbol{z}_j = f_a(f(\boldsymbol{x}_j, \boldsymbol{w})) \tag{3.15}$$

All the variables are shown in the Figure 3.6 and the entire network function with its various stages can be formulated as the following [Bis06]:

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = f_a(\sum_{j=1}^{M} \boldsymbol{w}_{kj}^{(2)} f_a(\sum_{i=1}^{D} \boldsymbol{w}_{ji}^{(1)} + \boldsymbol{w}_{j0}^{(1)}) + \boldsymbol{w}_{k0}^{(2)}) \tag{3.16}$$

During the training process in deep feed-forward neural networks, the aim is to reduce the disparity between $f^*(x)$ and the model function $f(x)$. For example, the disparity will be calculated based on the prediction result $f(x)$ and the label $f^*(x)$ by loss function or cost function.

Loss Functions for Classification

The choice of the loss function is one important aspect of neural networks. The principle of maximum likelihood is usually applied to calculating the distribution of prediction $p(y|x, w)$. Accordingly, the cross-entropy between the predictions and the labels of the model is defined as the loss function. This

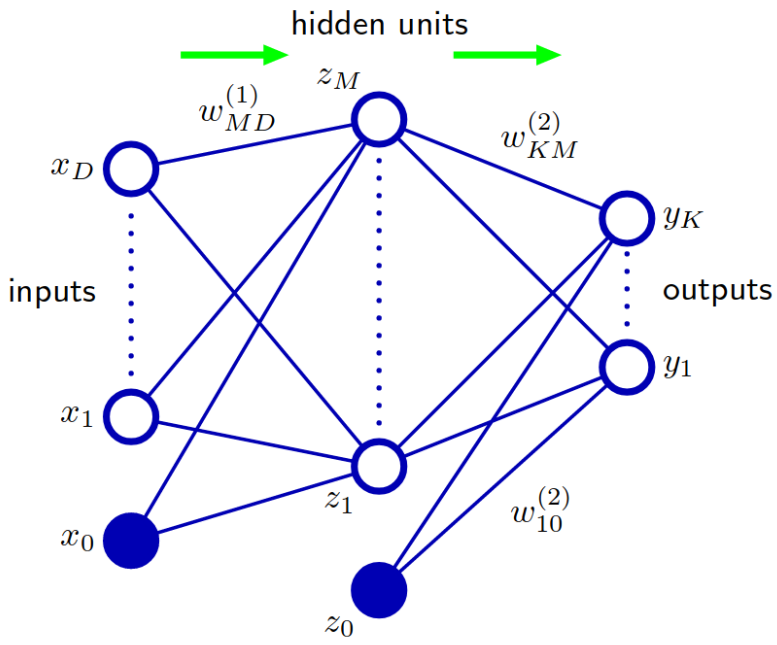


Figure 3.6: Network diagram with input, hidden, and output variables for a two-layer neural network [Bis06].

loss function is represented by the Equation 3.17 [GBC16]:

$$J(w) = -\mathbb{E}_{x,y\sim p_{\text{label}}} \log p_{\text{model}}(y|x) \tag{3.17}$$

Cross-entropy loss measures the performance of a classification neural network and the prediction result of the neural network is a value from 0 to 1. The value of the loss will increase as the discrepancy between the label and prediction result increases. The mathematical cross-entropy loss function can be formulated as the following:

$$J_{\text{cross entropy}}(w) = -\sum_{c=1}^{M} y_{\text{class}} \log(p_{\text{class}}), \tag{3.18}$$

where M means the number of classes and p means the predicted probability of the class.
Then the goal of the neural networks is to solve the optimization problem: [GBC16]

$$f^* = \text{argmin} J_{\text{cross entropy}} \tag{3.19}$$

Activation Functions

Non-linearities play an essential role in neural networks and the activation functions are often added after the network layers. The most commonly used activation functions aim to address non-linear problems. Table 3.1 presents the most frequently used activation functions and their mathematical expressions.

Table 3.1: Mathematical expressions of activation functions

Activation Function	Equation	Range
Sigmoid Function	$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$	$(0, 1)$
Hyperbolic Tanjant Function (tanh)	$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	$(-1, 1)$
ReLU	$f(x) = \begin{cases} 0; & x \leq 0 \\ x; & x > 0 \end{cases}$	$[0, \infty)$

Parameter Optimization

During the network training, our task was to find a weight vector w which minimized the chosen loss function $J(w)$. The most commonly used techniques will initialize a weight value w_0 and update it through weight space with the formulation:

$$w^{(t+1)} = w^{(t)} + \Delta w^{t} \tag{3.20}$$

where t means the training step and Δw^t is computed with gradient information.
Meanwhile, a parameter of learning rate η is defined to control the training speed. Finding an optimal learning rate plays an important role in the training process. For example, the gradient descent optimization is used to move the weight in the direction of the greatest rate of the minimization of the loss function (Equation 3.21).

$$w^{(t+1)} = w^{(t)} - \eta \bigtriangledown J(w^{t}) \tag{3.21}$$

where $\bigtriangledown J(w^t)$ points the direction of the best rate for increasing the loss function and η refers to the learning rate parameter, which are introduced in a later subsection [Bis06].
The backpropagation is a broadly algorithm utilized in updating the weight variables of feedforward neural works during network training. The loss function can measure to the errors of the prediction results and labels. Moreover, the backpropagation algorithm can calculate the gradient of the errors with

respect to the variables of the weight vectors, which is also called error backpropagation. In the next subsection, we explain how error backpropagation calculates the gradient of the loss function and update the weights during training networks.

Error Backpropagation

The error between the model result and the label are calculated with a loss function. Then the error $J(w)$ is managed with error backpropagation or backprop, with the aim of updating the variables w in the neural networks. In order to clarify the error backpropagation, the way in which the training process is accomplished should be noted. The most common training process for neural networks in this regard includes minimizing loss by adjustments to the weights in the networks [Bis06].
The gradient of the loss function with respect to one weight variable w_{ji} is calculated as $\frac{\partial J(w)}{\partial w_{ji}}$. In a feed-forward network, each neuron computes the weighted sum from the inputs as in Equation 3.15. The chain rule for partial derivatives can be used in the Equation 3.15 [Bis06] :

$$\frac{\partial J(w)}{\partial w_{ji}} = \frac{\partial J(w)}{\partial f(x_j, w_{ji})} \frac{\partial f(x_j, w_{ji})}{\partial w_{ji}} \tag{3.22}$$

At the same time, δ_j represents the first part of the unit of the output layer $\frac{\partial J(w)}{\partial f(x_j, w_{ji})}$ and z_i refers to the derivation of the output with respect to the weight $\frac{\partial f(x_j, w_{ji})}{\partial w_{ji}}$. Meanwhile, δ_j is computed with the partial derivatives of output unit δ_k with respect to the weights w_{kj}. We can compute the δ_j with the corresponding δ_k for every units in the output, as in Equatioin 3.23 [Bis06].

$$\delta_j = \sum_k \frac{\partial J(w)}{\partial f(x_k, w_{ki})} \frac{\partial f(x_k, w_{ki})}{\partial f(x_j, w_{ji})} \tag{3.23}$$

Then, the calculation of the whole pipeline for the hidden layer through backpropagation can be illustrated as in Figure 3.7 [Bis06]. Finally, the derivative of the total error/loss function $J(w)$ can be obtained with the following backpropagation formula:

$$\frac{\partial J(w)}{\partial w} = \sum_n \frac{\partial J(w)}{\partial w_{ji}} \tag{3.24}$$

Optimization Methods

Several improvements to the gradient descent, which was described in the previous subsection, are in frequent use today. In this subsection, we introduce the stochastic gradient descent (SGD) and Adam optimization methods, which were used in our work.
During training network, the deep learning aims to minimize a loss function, while the weight variables that minimized errors is to be calculated. Then, a SGD method was presented to minimize the loss functions as Equation 3.21 [GBC16]. The learning rate is required in the SGD algorithm. Firstly, the

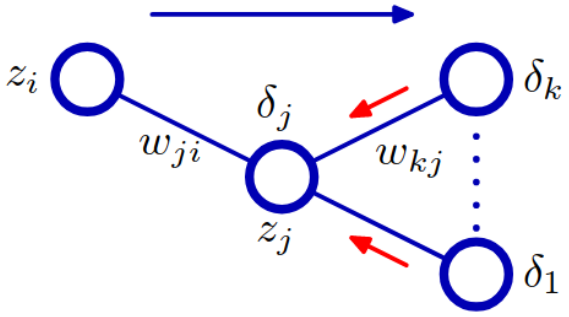


Figure 3.7: Illustration of the backpropagation pipeline for the hidden layer [Bis06].

parameters of weight vectors are initialized and then, the gradients of parameters are computed based on a minibatch of training dataset. Finally, the new parameters, which can minimized the errors, are updated as in Equation 3.21 [GBC16].
In order to perform updates with a higher speed, researchers introduced the momentum method, which enables the gradient to influence the speed of the optimization process. The rule of momentum rule can be presented in the following formulation. [RHW86]

$$\begin{aligned} \boldsymbol{v} &\leftarrow \mu \boldsymbol{v} - \eta \cdot \nabla_{\boldsymbol{w}} J \\ \boldsymbol{w} &\leftarrow \boldsymbol{w} + \boldsymbol{v} \end{aligned} \tag{3.25}$$

where the v means the velocity that accumulates through the gradient and μ refers to the momentum with the physical meaning of friction.
The Adam optimization method is a widely used adaptive optimizer, which combines the gradient's magnitude and its direction. The Adam optimizer is formulated as Equation 3.26. In addition, the learning rate parameter η affects the speed of optimization. [KB14]

$$\begin{aligned} c_1 &\leftarrow \alpha c_1 + (1-\alpha)\left(\nabla_w J\right) \\ c_2 &\leftarrow \beta c_2 + (1-\beta)\left(\nabla_w J\right)^2 \\ \boldsymbol{w} &\leftarrow \boldsymbol{w} - \eta \cdot \frac{\boldsymbol{c}_1}{\sqrt{\boldsymbol{c}_2 + \boldsymbol{\epsilon}}} \end{aligned} \tag{3.26}$$

where the α and β control the speed of decay. The ϵ is a smoothing variable for the gradient, to provide numerical stability.

Learning Rate and Learning Rate Decay

In the training period, the learning rate defines how quickly the network can be trained or optimized. Before training the network, the learning rate is defined according to the network structure and its parameters. If the network is trained with a high or low learning rate, the network weights will not be correctly updated. If a low learning rate is selected for training, the computed gradients in Equation 3.21 are too tiny to update weight variables quickly. In addition, training with a very high learning rate may lead to the updated weights are not able to extract optimal features from input dataset. Therefore, it is important to find the optimal learning rate. During the optimization of the network, the results of the loss function are reduced, which means errors between the prediction result and label decrease.
Meanwhile, learning rate decay is an effective technique within the optimization, in which the learning rate begins with a large value that decays in the training period. Empirically, learning rate decay helps with the optimization of the network. An initial large learning rate steps up the training with a high gradient and avoids the network facing the local minima problem, that is, that optimization with a low learning rate escapes the region with a local minimum rather than a global minimum. Secondly, decaying the learning rate helps the network to update without oscillation problem [YLWJ19].
Specifically, the exponential decay learning rate was used in our work. The learning rate can be described with the formulation:

$$\eta(t) = \eta(0) \cdot d^{-\frac{t}{k}} \tag{3.27}$$

Here t refers to the training step, $\eta(0)$ means the initial learning rate, d means the decay rate, and k is the decay step.

3.2.2 Hyperparameters

During training networks, the hyperparameters are the properties in the whole process, such as learning rate, number of training iterations and activation functions. In this subsection, some of the hyperparameters used during the CNN training process are briefly introduced.

- Batch Size: During the training process, only a limited number of inputs can be fed into the network and the dataset is divided into a number of batches. As a consequence, the number of one mini-batch is the batch size, which is defined before the training of the network begins.
- Iterations: The iteration is the number of training mini-batches.

3.2.3 Convolutional Neural Networks

The most important layers used in CNNs are the fully connected layer, the convolutional layer, and the pooling layer. A simple CNN architecture is depicted in Fig. 3.8.

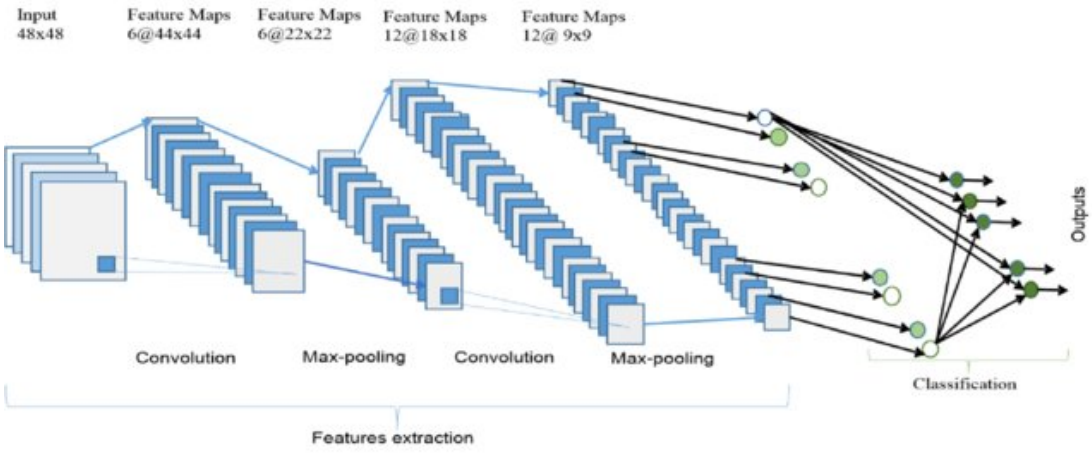


Figure 3.8: An simple CNN Architecture with the fully connected layer, the convolutional layer, and the pooling layer[ATY+19].

The main functions of the different layers can be detailed as follows:
The convolutional layer calculates the output of neurons, which are in the local regions of the input. The weights in the layer refer to the parameters during the calculation. Meanwhile, the activation functions, such as ReLU and tanh, are applied after the convolutional layers.
The pooling layer downsamples all of the given results of the activation functions, while the number of parameters is reduced. The fully connected layer can perform functions similar to the multilayer perceptron, which can estimate the possibility map for classes from the output of the last activation function to be used in the classification task.

Convolutional Layer

Specifically, the convolution operation can be formulated as in Equation 3.28 [GBC16]. The convolutional layer is operated using learnable kernels, which have a receptive field. Based on the convolutional operation, the learnable kernel is convolved with the input data, while the kernel filter is slid over the input for spatial dimensions. If we define the output feature map as S, the convolution operation for image input I can be depicted with formulation 3.29. An simple example of 2D convolution is shown in Figure 3.9 [GBC16].

$$s(t) = \int x(a)w(t-a)da = (x * w)(t) \tag{3.28}$$

Where $w(a)$ means the weighting function, x(a) refers to the input, and t means time.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{3.29}$$

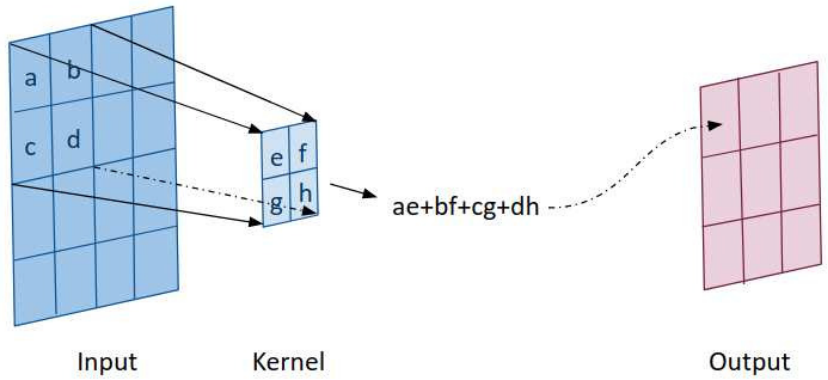


Figure 3.9: An example of 2D convolution. The boxes with arrows indicate how the upper-left local features of the output are calculated by applying the kernel to the corresponding upper-left region of the input data.

Convolutional layers are not equal to the basic transformation methods, such as the rotations and scales of an image. They can extract feature maps from the input data, which simulate the calculations of artificial neurons.

Pooling Layers

Pooling layers are often used after convolutional layers, where the output of the layer is modified with a pooling function. For example, max polling functions are able to calculate the maximum outputs in a rectangular region of the layer output. Meanwhile, there are also other pooling functions that can calculate the average value of the rectangular region [GBC16]. The pooling layer can reduce the spatial size of feature maps, in which the number of parameters decreases.

Batch Normalization

Batch normalization [IS15] is a popular operation often used in CNNs. A recognized problem within training neural networks is internal covariate shift, in which the updating of weights in the layers close to the output also influences the weights in the lower layers and the updated weights in the lower layers worsen. For this reason, a low learning rate has to be selected during the training process.
To address the internal convariate shift problem, the layer inputs are normalized by a batch normalization layer. A batch normalization layer with d-dimensional input x can be expressed with the formulation:

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathbb{E}[x]}{\sqrt{\mathrm{Var}[x] + \varepsilon}} \tag{3.30}$$

where $\mathrm{E}[x]$ means the mean of samples and $\mathrm{Var}[x]$ refers to the sample variance. Meanwhile, ε is a small constant used to prevent divison by zero [IS15].
In practice, the batch normalization layer can be added after the convolutional layers. The normalized results are fed into activation functions in order to address non-linear problems. During our work, we used the ReLU and tanh activation functions in our framework.

3.2.4 Network Architectures

Since AlexNet [KSH12] produced a remarkable performance in the ImageNet classification task in 2012, deep learning methods have been widely used in different challenging tasks. Inspired by AlexNet, many CNNs have been presented in the last few years, such as VGG[SZ14], GoogLeNet[SLJ+15], and ResNet [HZRS16a] and its variants[HZRS16b]. When all of the popular CNN's are compared, the latest residual network (ResNet) and its variants achieve state-of-the-art performances for different benchmarks, including the object classification on ImageNet, object detection and even the segmentation task in the COCO dataset [HZRS16a]. In the following subsection, the details of ResNet that were used in our work are detailed.

ResNet

ResNet provides a network with blocks of residual learning, which is shown in Fig. 3.10. The ResNet is trained based upon building blocks of residual learning and could achieve high accuracy [HZRS16a].

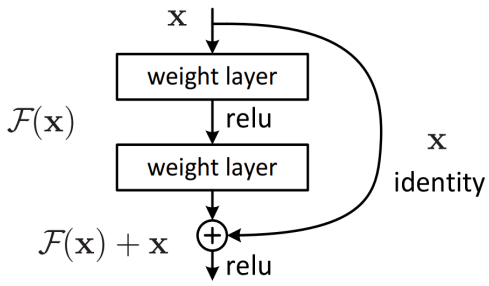


Figure 3.10: A Building Block of Residual Learning [HZRS16a].

The deep residual learning framework can address degradation problems during the training of a network. Formally, the desired result $H(x)$ is optimized with weight layers based on the condition: $H(x) = F(x) + x$. In this way, ResNet fits the identity x with the weighted non-linear layers, which is easier than optimization with the original method. Meanwhile, the formulation $F(x) + x$ enables that one or more layers to be skipped in the feedforward neural networks, which are called shortcut connections [HZRS16a]. Following this principle, ResNet-34 and ResNet-50 were able to be built. The main network architecture of Res-Net 34 is shown in Fig. 3.11. Through experiments, the ResNet has

proved that it is able to solve the accuracy reduction problem when increasing the number of layers, that is , the accuracy decreased during increasing the number of layers as well as number of weights.

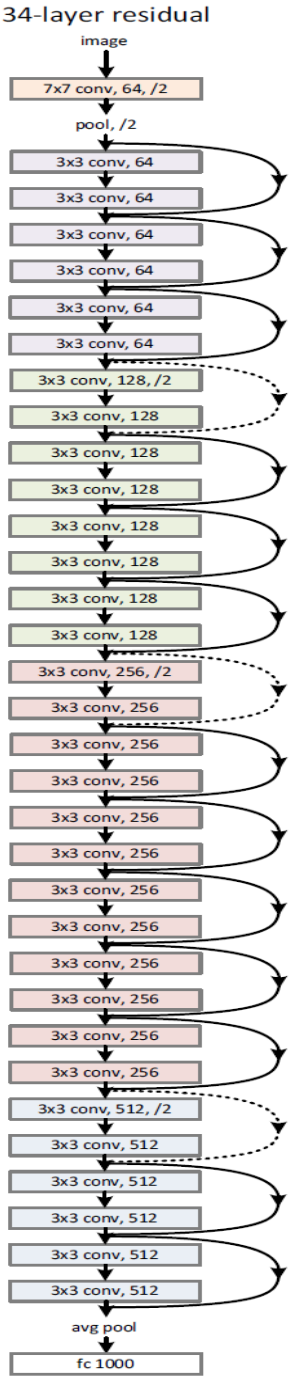


Figure 3.11: The network architecture of ResNet-34 [HZRS16a].

Subsequently, He et al. presented different variants of ResNet, in which they analyzed the propagation formulations behind the residual blocks and investigated the different variants of the residual building structures, as shown in Fig. 3.12. Finally, the variants in Fig. 3.12 (a) and (e) were labeled ResNet v1 and ResNet v2 [HZRS16b].

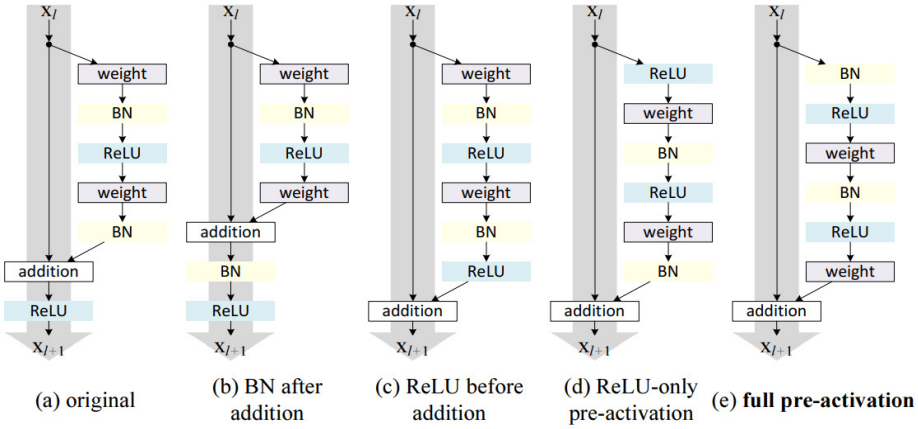


Figure 3.12: Variants of the residual building blocks [HZRS16b].

3.2.5 Transfer Learning

Transfer learning is the aptitude of a learning algorithm to research transferability between different learning tasks or datasets, with the aim of sharing the weights in the functions and transferring knowledge across domains [Ben13]. In some fields of robotics and bioinformatics, building a large dataset with reliable human annotations is a difficult task because it is time-consuming and expensive. In this subsection, the main pipeline of transfer learning methods is outlined. Firstly, it is assumed that the task contains label space and a target prediction function. In addition, the domain definition is used to describe the feature distributions. Fig. 3.13 shows the simple pipeline of the transfer learning method.

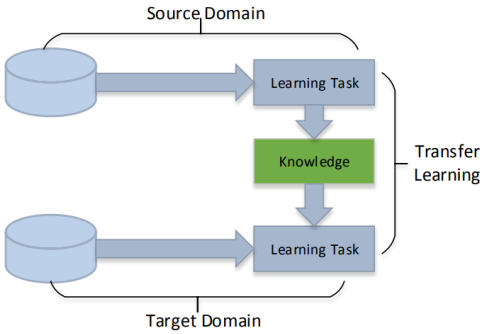


Figure 3.13: The pipeline of transfer learning [TSK+18].

Before learning a task T_t based on the domain D_t, the knowledge from the trained function of task T_s with source domain D_s can be utilized in the initialization period. Transfer learning could improve the performance of the function for task T_t based upon the knowledge from D_s and T_s, which are very helpful for tasks without a large reliable dataset [TSK+18]. During our work, we utilized the knowledge from the object recognition task in ImageNet dataset [DDS+09] to train our image-based robotic grasping framework.

3.3 Basic Concepts of the Data Preprocessing

3.3.1 Depth-jet Encoding Method

There are many methods for color encoding of depth images, which encode the original depth data to color maps. Fig. 3.14 depicts some popular depth encoding methods broadly used in computer vision [ESS+15], such as depth-gray, depth-jet encoding and HHA.

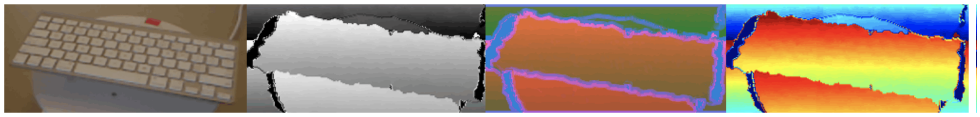


Figure 3.14: Popular methods for color encoding of depth images [ESS+15]. From left to right: RGB, depth-gray, HHA, depth-jet encoding.

Depth-jet encoding is a encoding approach, which transfer the depth data with a range of [0, 1] to a three-column array. In addition, the depth values are encoded into various colors. Figure 3.15 depicts a illustration of the depth-jet encoding method. The depth values are encoded into different colors. Finally, the depth-jet colormap can be represented as the right image in Fig. 3.14.

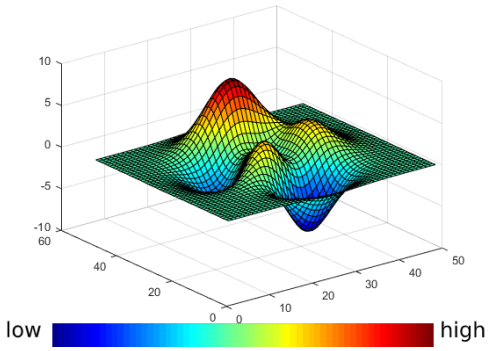


Figure 3.15: Illustration of depth-jet encoding [Mat].

The HHA encoding method encods the original depth data into three channels, such as the height above the ground, the horizontal disparity and the angle between the normal orientation and the gravity direction [ESS+15]. Researchers investigated the influence of the depth encoding methods on the result of CNN for object recognition task. Table 3.3 presents the results of object recognition task based upon depth-gray, HHA and depth-jet encoding methods. In conclusion, the CNN with input image based upon the depth-jet encoding method achieved higher performance than the CNNs based upon depth-gray encoding and HHA method. In our robotic grasping framework, we trained the CNN based upon transfer learning method, while the pre-trained model was utilized in extracting feature maps from the input images. Following this, the depth-jet encoding method was used to encode the original depth data to jet color maps during our robotic grasping framework due to the high performance of the depth-jet colormap in [ESS+15].

Table 3.3: Comparison of different depth encoding methods utilized in object recognition [ESS+15].

Depth Encoding Methods	Accuracy
Depth-gray (single channel)	80.1 ± 2.6
Depth-gray	82.0 ± 2.8
HHA	83.0 ± 2.7
Depth-jet encoding	83.8 ± 2.7

3.3.2 Gaussian Distribution

Because Gaussian distribution method was utilized in generating our dataset for robotic grasping, the basic concepts of Gaussian distribution are introduced in this subsection.
Gaussian distributions were often used in the probability theory and it is a kind of continuous probability distribution of a random parameter. The Gaussian distribution function can be formulated [Wik20]:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \quad -\infty < x < +\infty \tag{3.31}$$

where μ refers to the mean of the distribution and σ means the standard deviation.
In addition, standard normal distribution is a simple example of Gaussian distribution, in which $\mu = 0$ and $\sigma = 1$ [Wik20]. Specifically, the Gaussian distribution functions were also used in robotic grasping [LZC+19]. Lin et al. presented a process of the grasp planning based upon Gaussian distribution. The grasp samples were estimated based upon two dimensional Gaussian distribution, as shown in Figure 3.16 [LZC+19]. Following this, we designed a data generation method based upon the Gaussian distribution-based functions and grasp measure methods.

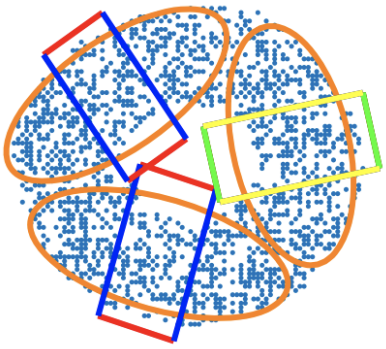


Figure 3.16: Illustration of Gaussian distribution in robotic grasping [LZC+19].

3.4 Conclusion of Foundations

In this chapter, the basic concepts of the contact model were introduced, such as force-closure grasp and grasp measure method. The force-closure metric was utilized in sampling the grasp candidates based upon the basic definition of friction cones during the data generation for robotic grasping. After obtaining all the grasp samples, the grasp measure method - Ferrari-Canny metric - was used to estimate the grasp qualities. The grasp central positions and the corresponding grasp qualities were labeled in the label maps. In particular, one problem was that there are no labels in pixels between the contact points and grasp central points. Following this, we designed a Gaussian distribution-based function for calculating the grasp qualities between the contact points and the grasp central points using the grasp qualities of the grasp central points. Then, we built an image-based dataset for robotic grasping. Specifically, the input data was the depth-jet encoded colormap due to the high performance of depth-jet encoding method in object recognition. The grasp central points with grasp qualities estimated by the grasp measure method as well as the pixels between grasp centers and contact points with the grasp qualities, which was calculated based upon the Gaussian distribution functions, were labeled in label maps. With the synthetic dataset, we designed a kind of encoder-decoder network for robotic grasping, which consisted of many parameters. During the training process, we investigated the influence of activation functions, loss functions, and optimizers on the training procedure. Finally, our deep learning-based robotic grasping framework was built based upon the encoder-decoder network.

4 Method

The following sections outline the methods utilized in our deep learning-based framework for robotic grasping, which consisted of the following:

- Characterizing the data generation model for robotic grasping and ascertaining the appropriate model for this pipeline.
- Designing a deep learning-based framework for robotic grasping.
- Training the designed network with different generated datasets.

4.1 Problem Statement

We considered the problem of building a robotic grasping framework for a large sum of unknown objects. The collaborative robot was equipped with a two-jaw-gripper as the end-effector and an RGB-D camera, which observed the state of the scene as an RGB image $\boldsymbol{I}^{\mathrm{rgb}} \in \mathbb{R}^{640\times480\times3}$ and a depth image $\boldsymbol{I}^{\mathrm{d}} \in \mathbb{R}^{640\times480\times1}$.
The goal of the thesis was to propose a synthetic grasping data generator that quickly generated the datasets for evaluating image-based grasping utilizing analytical methods and building an agile robotic grasp framework (AgileGrasp-Net) for a robot that could reiteratively grasp unknown objects in a real-world environment based on the depth image.
The synthetic grasping data generator observes the state of object model $\boldsymbol{x}^{\mathrm{model}}$, including the mesh, the stable pose on a planar surface and the material properties. The stable poses $\boldsymbol{u} = p^{\mathrm{stable}}(\boldsymbol{x}^{\mathrm{model}})$ of the object mesh on the surface were derived from geometric properties. Based on the conditions of force-closure (see Section 3.1.3), a grasp sampler with a pose filter was utilized to obtain all of the horizontal force-closure grasp poses $\boldsymbol{v} = \pi_\theta^{\mathrm{fc}}(\boldsymbol{u}), \boldsymbol{v} \in \mathbb{R}^{4\times4}$ of a two-jaw-gripper, where θ refers to the parameters in the contact model. A grasp pose was defined as a 3D position with the 2D orientation of the end-effector. Behind the grasp quality calculation based upon grasp measure metric (Ferrari-Canny metric), the grasp quality $\boldsymbol{w} = \pi_\theta^{\mathrm{g}}(\boldsymbol{v})$ was entrusted to every grasp sample. Finally, the grasp scenes were rendered in the pyrender environment, while the camera configuration θ_{c} and the volume of object θ_{v} were randomly generated according to the arrangement in the real world. The synthesized images were defined as $\boldsymbol{y} = R(\theta_{\mathrm{c}}, \theta_{\mathrm{v}})$. We also used the multiprocessing to accelerate the data generation process. The possibility of the image output corresponding to the states and parameters of the data generator was calculated using the following formulation:

$$p(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w},\boldsymbol{x},\boldsymbol{y},\theta,\theta_{\mathrm{c}},\theta_{\mathrm{v}}) = \underbrace{p(\boldsymbol{y}|\boldsymbol{w},\theta_{\mathrm{c}},\theta_{\mathrm{v}})}_{\text{renders}}\underbrace{\pi^{g}(\boldsymbol{w}|\boldsymbol{v},\theta)}_{\text{grasp quality}}\underbrace{\pi^{\mathrm{fc}}(\boldsymbol{v}|\boldsymbol{u},\theta)}_{\text{force-closure}}\underbrace{p(\boldsymbol{u}|\boldsymbol{x}^{\mathrm{model}})}_{\text{stable pose}}\underbrace{p(\boldsymbol{x}^{\mathrm{model}})}_{\text{state}} \tag{4.1}$$

According to the data generator, we generated the image-based synthesized data $\boldsymbol{I}^{\text{synthesized}}$ and the corresponding inference map with a four degrees-of-freedom (4DOF) grasp pose$p^{\text{label}} = (\boldsymbol{x}_{\text{grasp}}, \boldsymbol{y}_{\text{grasp}}, \boldsymbol{z}_{\text{grasp}}, \theta_{\text{grasp}})$ as the training set for AgileGrasp-Net. Moreover, the grasp quality in the inference map will be processed with Gaussian distribution in the grasp direction θ_{grasp} and the data augmentation of 16 rotations based on the grasp direction θ_{grasp} was employed in the inference maps. This resulted in every inference map referring to one label map for the grasps, whose orientation coincided with the rotation number. The AgileGrasp-Net was developed to learning the robotic grasp policy $I^{\text{InferenceMap}} = \pi^{\text{AgileGrasp}-\text{CNN}}(I^{\text{synthesized}})$. Finally, the experiments based on a real-world robot were executed according to the robotic grasp policy and the different evaluation metrics were measured by the grasp success rate of a physical robot.

4.2 Objects and Database

In order to train a flexible and robust CNN for robotic grasping, it's important to generate a large dataset in a simulation environment. For this purpose, we collected 1,521 commonly used object models from 3dNet [WARV12], the Yale-CMU-Berkeley (YCB) Dataset [CSB+17], Princeton ModelNet [WSK+15], Dexterity Network's (DexNet's) intergrated data models [MLN+17] [MMS+19], and the MVTec Industrial 3D Object Detection Dataset (ITODD) [DUB+17] as a grasping model database. The visualization of some of the selected models is shown in Figure 4.1.

Figure 4.1: Some of the models selected for the synthetic dataset.

Meanwhile, all the models were divided into different shape categories, such as spherical-like shapes, cuboid-like shapes, cup-like shapes, and complicated shapes. The grasping model database was utilized to generate grasping image datasets in the pyrender simulator and the labels were robustly and effectively calculated [Mat19].

4.3 Grasp Generation in a Simulation Environment

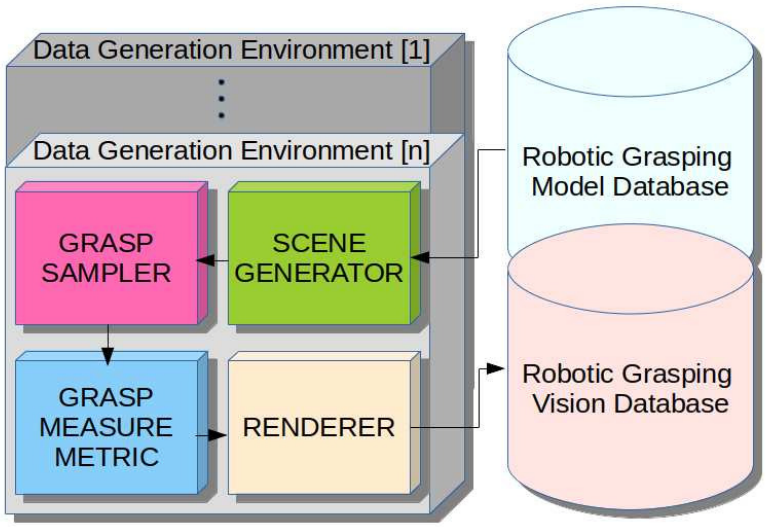


Figure 4.2: Illustration of data generation for grasping.

Utilizing recent developments in the analytical methods involved in grasping, these methods, including force-closure and the Ferrari-Canny grasp quality measure metric, were utilized to generate the synthesized images with object models according to the sensor configuration. Considering the motion planning of robots and the form of the sensor data, the robotic grasp framework was designed for grasps with a two-jaw gripper that were parallel to a planar table. Then the horizontal grasps with grasp qualities were generated with respect to the object.
The overall data generation system consisted of four parts (see Fig. 4.2):

- the scene generator that generated the poses of the object and table.
- the grasp sampler that received the scenes and generated grasp-point pairs considering the pose of the object and table, and the force-closure policy $\pi^{\text{fc}}(\boldsymbol{v}|\boldsymbol{u}, \theta)$.
- the grasp measure metric that calculated the grasp qualities of the force-closure grasps.
- the pyrender simulator responsible for rendering the images.

Note that the scene generator and pyrender simulation environment were virtual, and their speed depends upon the computation resources available in the central processing unit (CPU). Multi-threading models were utilized in the data generation process, with the apparent aim of acquiring more synthesized data in the unit time.

4.3.1 Grasping Data Generator

Similar to much of the related literature on robotic grasping, the grasps perpendicular to the planar surface were considered to be the solution for grasping based on the depth image of the scene (see Fig. 4.3). In this subsection, we describe the grasp data generator that used multifarious 3D models based upon force-closure and the grasp quality metric - the Ferrari-Canny metric - to generate the grasp scenes and calculate the probabilities of successful grasps.

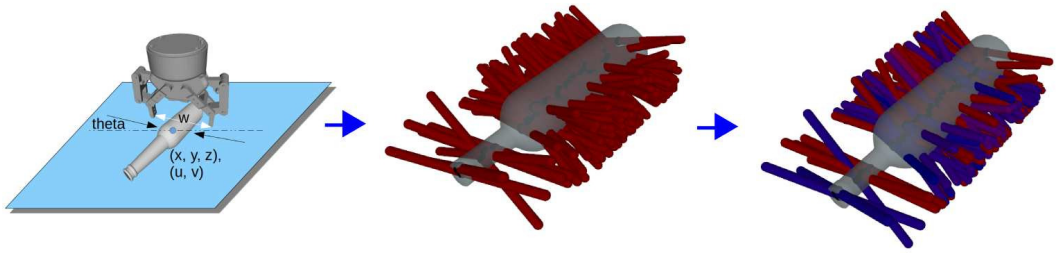


Figure 4.3: Illustration and generation pipeline of horizontal grasp. (Left) The grasp was sampled based on the 3D mesh model and table. (Middle) Only grasps whose orientations were perpendicular to the table were considered during the grasp generation. (Right) Each grasp was quantified with a value between 0-1 based on a robust analytical grasp quality metric (red: grasp with high-quality value, blue: grasp with low-quality value).

Horizontal Grasp Sampling

The grasp sampler is a function that oversamples the grasps g given the meshes of the object and one table as well as many parameters, including the friction coefficient, the threshold of the angle between the grasp direction and horizontal axis, and the others. Next, we discuss how the force-closure metric was applied to generate grasp candidates with the 3D model.

First, we assumed that the mass distribution of the object that was grasped was uniform. Then, the centroid can be easily calculated by means of shape representation, which was utilized to calculate the wrenches on the contact points. In order to maintain the success of the following computational analysis, watertight reparation was utilized at the outset with the functions from trimesh python library. At the same time, the volume was randomly generated in the range $[27\text{cm}^3, 1000\text{cm}^3]$, which was selected based upon the volume distribution of daily household items. Then, the stable poses of the object were generated given the meshes of the object and table. For every stable pose, the probability of the grasps $p(\boldsymbol{g}_{\text{sampling}})$ could be written as the following:

$$\begin{aligned} \boldsymbol{v} &= \pi_\theta^{\text{fc}}(\boldsymbol{u}), \boldsymbol{v} \in \mathbb{R}^{4\times4}, \\ p(\boldsymbol{g}_{\text{sampling}}) &= p(\boldsymbol{v}|\boldsymbol{u}, \theta)p(\boldsymbol{u}|\boldsymbol{x}^{\text{model}}) \end{aligned} \tag{4.2}$$

The grasp points were sampled on the surface of the object mesh with reference to the stable pose of the object model and the grasp direction was selected in the friction polyhedral convex cone, which is the condition of force-closure. During the sampling of multiple grasp points, the distances between every two contact points were considered. If the distances are more than the maximal width of two-jaw gripper, the corresponding grasp samples were deleted.

The Ferrari-Canny metric, also called the epsilon metric, was utilized in every grasp sample to evaluate the grasp quality. The main outline of the Ferrari-Canny metric is shown as Algorithm 1. The algorithm considers the frictions, torques and normal forces in the contact model to build the wrench space. According to the convex hull of the wrench space, in which the forces and torques possibly executed on the contact points could be estimated, the weakest point of the wrench was examined with the quadratic programming method (used in non-linear programming) and the ϵ parameter was expressed as the scaled distance from the corresponding point on the convex hull of the wrench space to the origin point.

Algorithm 1: Ferrari-Canny metric policy $\boldsymbol{w} = \pi_\theta^{\mathrm{g}}(\boldsymbol{v}_j)$

Given a force-closure grasp $\boldsymbol{v}_j$ and contact model parameters $\boldsymbol{x}^{\mathrm{contact}}$;
Map grasp to friction polyhedral convex cone with m edges.;
Calculate contact model containing of frictions, torques and normal forces;
Build convex hull $\boldsymbol{G}$ in wrench space $\mathbb{R}^6$ based on the contact model ;
for *facet in convex hull* $\boldsymbol{G}$ **do**
    Solve quadratic programming problem $\min_{\boldsymbol{x}\in\mathbb{R}^6} \boldsymbol{x}^T \boldsymbol{G}\boldsymbol{x}$;
end
Find the minimum epsilon parameter as grasp quality;

Algorithm 2: Grasp Data Generator $f(x)$

Given 3d models $\boldsymbol{x}^{\text{model}}$ (object and table) and contact model parameters $\boldsymbol{x}^{\text{contact}}$;
Initialization for mesh volume and reparation for watertight;
for *number of multi-threading* **do**
    Load mesh from database in each threading;
    Fill holes, fix normals of meshes, repair the watertight of meshes;
    while *mesh is watertight* **do**
        Sample the volume of object in range of $27 - 1000\text{cm}^3$;
        Infer stable poses $\boldsymbol{u}$ = trimesh.compute_stable_poses($\boldsymbol{x}^{\text{obj}}$,$\boldsymbol{x}^{\text{table}}$);
        for *stable pose* $\boldsymbol{u}_i$ **do**
            Sample grasps $\boldsymbol{v} = \pi_\theta^{\text{fc}}(\boldsymbol{u}_i)$, which are perpendicular to the planar surface according to the conditions of force-closure (Section 3.1.3);
            for *grasp* $\boldsymbol{v}_j$ **do**
                Calculate grasp quality $\boldsymbol{w}_j = \pi_\theta^{\text{g}}(\boldsymbol{v}_j)$ using ferrari-canny metric;
            end
        end
        Store mesh, stable pose, grasp pairs and corresponding grasp qualities to HDF5 files $\boldsymbol{F}_{\text{grasp\_sampling}}$;
        Analyse distribution of grasp qualities for every model;
    end
end

According to the grasp generator, the list of force-closure grasps with grasp qualities was appended to the database. Algorithm 2 describes the main pipeline of the grasp data generation. Finally, the database was saved as HDF5 files and will be utilized to generate training samples for the network.

4.3.2 Generation of Training Samples

In order to generate a large number of training samples for the deep learning-based method for robust robotic grasping with a 3D sensor, the visual model with the camera settings and an object pose above one table were simulated in a pyrender simulation environment. This process was called scenario generation. The camera configurations were randomly generated based upon the configurations of the physical depth camera. In addition, the parameters of camera pose were randomly generated. The distance between the camera and the table was randomly selected in the range [65mm, 75mm] during rendering. Figure 4.4 (Left) shows the parameters of camera model. After generating the pose of camera, the object model and one planar table model was fed into the pyrender simulation environment. Firstly, the volume of the object was randomly sampled in a range of $27 - 1000cm^3$, which encompassed the sizes of commonly used household objects. In addition, the implementation details in the scenario generation are shown in Table 4.1. Then, the stable pose of object on the planar table was estimated based upon the function from trimesh python library. After volume randomization and the calculation of

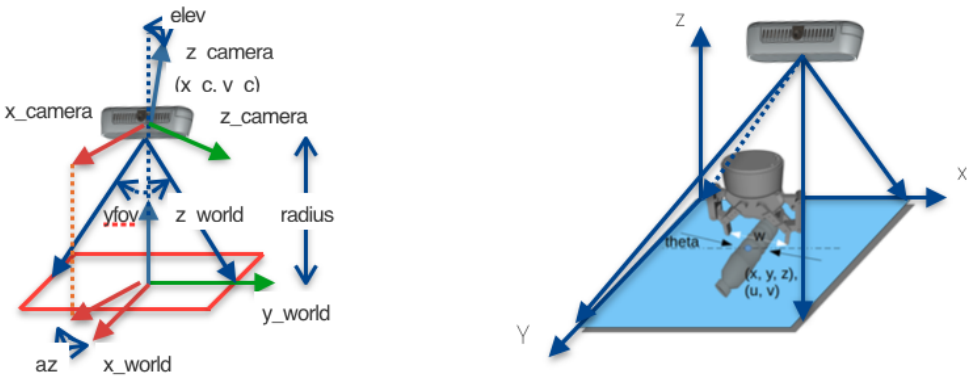


Figure 4.4: Pipeline for scenario generation. (Left) The parameters of camera were generated. (Right) After generating the camera model, the visual model consisting of object poses was generated.

object's stable pose, we sampled the horizontal force-closure grasps $\boldsymbol{g}_{\text{sampling}}$ by taking consideration of the orientation of the contact model. During sampling force-closure grasps, one grasp with two contact points in the object surface was sampled taking consideration of gripper width. The contact points with a larger distance than the maximal gripper width were ignored. After obtaining two valid contact points of one grasp, the normal directions of two contact points were estimated based upon the 3D model. Then, the grasp direction can be calculated with the two contact points. Specifically, the grasps parallel to the planar table surface were selected based upon the angle between the grasp direction and the z-axis of world coordinate system. Finally, we sampled a set of grasp samples, which were parallel to the planar table. Figure 4.4 (Right) presents the visual model and the sampled horizontal grasp sample and the grasp sample can be described with the position of grasp central point (x, y, z) in world coordinate system or (u, v) in image coordinate system, the angle θ between grasp direction and x-axis, and the distance between contact points.

Specifically, we sampled 150 horizontal grasps for one object in one stable pose, which was perpendicular to the table. Stable poses without enough grasp samples were ignored. For the sake of quantifying the grasp quality, we applied the Ferrari-Canny metric policy to the object of every stable pose on the planar table. Fig. 4.3 and the top left of Fig. 4.5 demonstrate the generated horizontal grasps with different colors. The grasps in red had higher grasp qualities than the samples in blue based upon the grasp quality metric. Each grasp sample was staged with one grasp central point and one orientation. The central points of all grasp samples were labeled in the synthetic image coordinate systems after distributing the orientations of grasp samples into 16 groups, in order to avoid predicting the orientation of grasp during network inference. Firstly, the entire grasp qualities of the object in different stable poses in one visual model were normalized to a range of 0-1. Figure 4.5 depicted the whole pipeline of distributing the grasp samples to 16 rotated depth-jet colormaps. After normalization of grasp qualities of all the grasp samples, the threshold was selected with the median value of the grasp qualities. The grasp central points in label maps were labeled three classes, such as negative grasp with RGB value

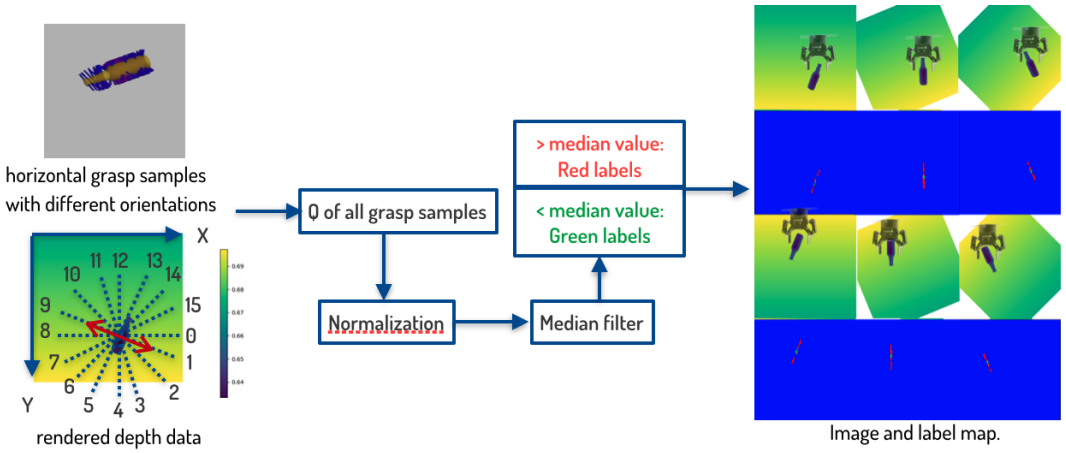


Figure 4.5: Pipeline for distributing all the grasp samples into 16 orientation zones. Specifically, the grasp samples were divided into 16 groups according to the grasp directions and the grasp central positions were labeled in the corresponding rotated depth-jet colormap. Following this, the pixels in label map can mean the possibility of grasps parallel to the x-axis of image coordinate system. When taking consideration of grasp qualities of all the samples, the pixels of all the grasp samples in label map were labeled in red or green. The red pixels refers to the grasp parallel to x-axis of image coordinate system and with lower grasp qualities. In addition, the green means the second class, which consists of the grasps with higher grasp qualities, which are parallel to x-axis of image coordinate system. The pixels of background were labeled in blue.

of [1, 0, 0], positive grasp with RGB value of [0, 1, 0], and background with RGB value of [0, 0, 1] in image. The grasp central positions with grasp qualities, which were higher than the threshold, were labeled in green and these pixels belonged to class of positive grasps. In addition, the grasp central points in red, which locate in label maps, refers to grasp samples with lower grasp qualities and the class of negative grasps. In addition, we distributed the grasp samples into 16 groups according to the angles between grasp directions and x-axis of image coordinate system, as shown in bottom left of Fig. 4.5. Then, the synthetic depth-jet colormaps were rotated 16 times according to the 16 groups in order that the pixels in label maps presented the grasp samples, which were parallel to the x-axis of image coordinate system. Following this, the pixels means the possibility of the three classes - negative grasps, positive grasps and background. Specifically, there are still the pixels in the object, which are not grasp samples. These pixels were labeled in black with RGB value of [0, 0, 0].

Figure 4.6 shows the whole pipeline of grasp database generation. We rotated the original depth image with 16 different multiples of $\frac{\pi}{8}$, with the aim of distributing all the grasp samples among 16 orientations. The pixel of the grasp center with the grasp direction, which was parallel to the x-axis of the rotated depth image, was fitted with the calculated grasp quality. According to these grasp qualities, we obtained

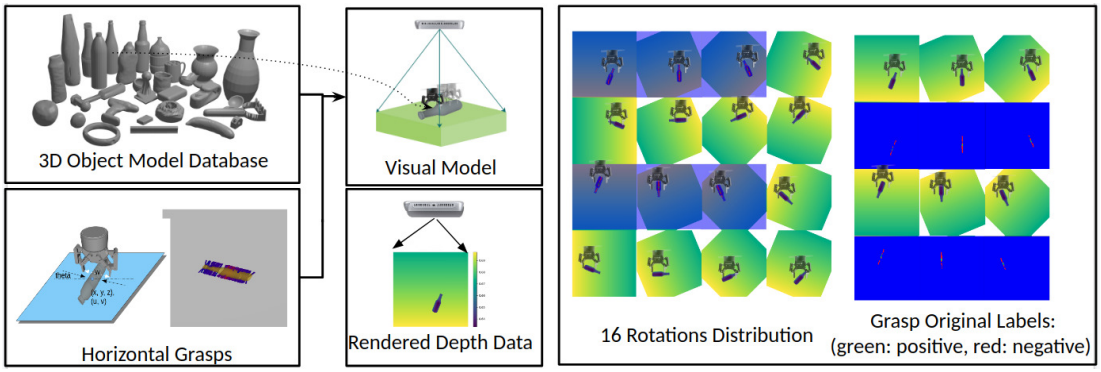


Figure 4.6: Pipline for grasp database generation.(Top Left) The 3D model database contained 1,521 3D object mesh models. (Bottom Left) For each model in one stable pose above the planar table, 150 force-closure grasps were sampled in the orientation, which was parallel to a planar table to maintain a uniform data distribution of labels in the training dataset; the grasps were quantified with robust analytic grasp metrics. (Middle) The point clouds of each object in the stable pose were rendered with the randomization of the object pose and camera configurations. In the rendered depth image (size: 600×600), each grasp was labeled with one pixel position and one orientation. (Right) For each stable pose of the object, the depth image was rotated between 16 different angles and all the grasp labels were distributed in 16 rotated depth images according to orientation for the purpose of keeping the grasp direction in line with the x-axis of the depth image. Finally, all the grasp qualities of each object were classified into two classes: positive and negative grasps. The central points of the grasps were labeled in the label map (positive grasp central points: green, negative grasp central points: red, pixels of background: blue).

quality label maps, which were created by labelling the quality in the corresponding pixel of the rotated depth image based upon the grasp central points and corresponding grasp qualities. The rotated depth images, in which there were no grasp samples were not considered in the generated datasets. Furthermore, in order to classify the generated grasps, the corresponding grasp samples were divided into two classes (positive grasps and negative grasps) based upon their qualities. We defined the samples whose grasp qualities were in the highest 50% of the quality metric as positive grasps and the remaining grasps as negative grasps. These are labeled in green and red in label maps while the blue indicates background. Following this, during training network, the robotic grasping task can be transferred to the classification task with the three classes- negative grasps in red, positive grasps in green and background in blue. The three channels of the predicted label maps refers to the possibilities of different three classes.

4.3.3 Data Post-Processing and Augmentation of the Training Samples

It is well-known that the possession of a substantial amount of data plays an important role in developing CNNs. To improve the quantity and diversity of the data, we generated our final grasping dataset with our existing database based upon data augmentation. We encoded the original depth data to depth-jet colormaps given that the depth data could be transferred to image data and the trained CNN could be insensitive to the sensor height above the grasped objects. The encoded jet colormaps were fed into the training network. In addition, we designed a novel Gaussian distribution-based data augmentation method for our database. The label maps with the Gaussian distribution-based data augmentation were treated as labels of CNN. The encoded depth-jet colormap and the Gaussian distribution-based data augmentation method are introduced in this subsection.

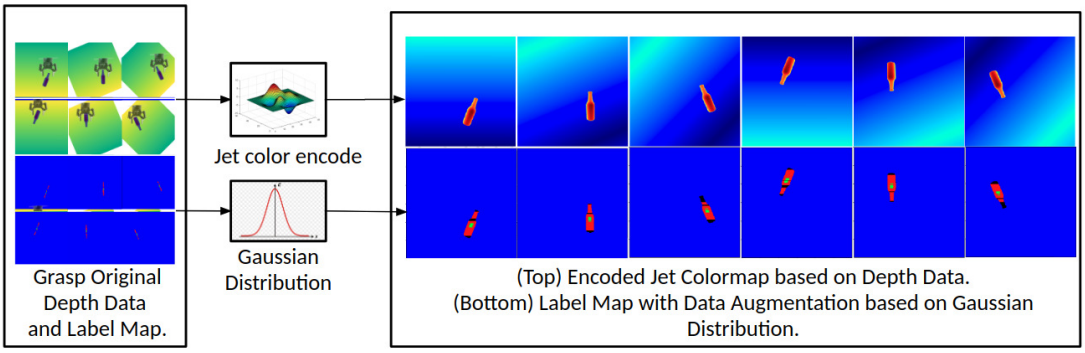


Figure 4.7: Post-processing of generated grasp data. (Right) Each depth image was encoded into one depth-jet colormap. The grasp quality value of the pixels in the area around the center of each grasp was estimated using the Gaussian function, which was fitted based upon the position of the grasp contact points. The full image dataset contains over 65k images.

Encoded Depth-jet Colormap

Color encoding is a fundamental way of representing scalar values in visualizations. In order to scale the depth data, we chose depth-jet color encoding as the post-processing method. The original depth data were firstly normalized to values of 0-1. The highest position in the scene was scaled to value of 1 and was encoded as the red with RGB value the [255, 0, 0]. Meanwhile, the lowest point appeared in blue in the encoded depth-jet colormap.

Data Augmentation of Grasp Qualities based on the Gaussian Distribution-based Function

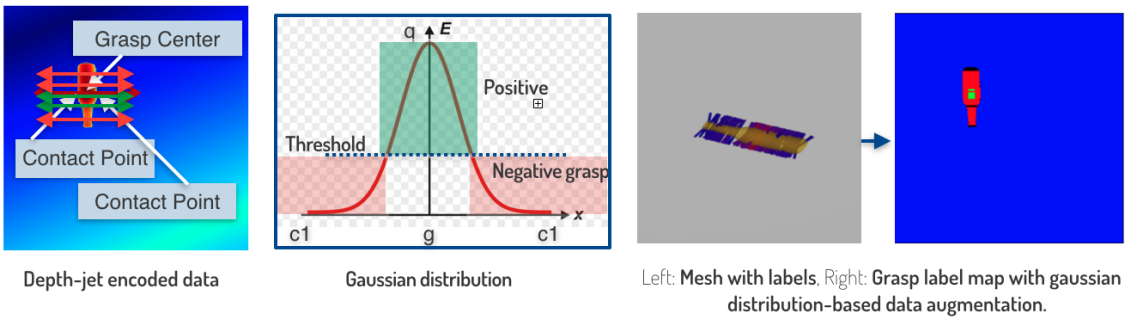


Figure 4.8: Pipeline of the Gaussian distribution-based data augmentation.

Taking into consideration of the quantity and diversity of the generated grasps, we designed a Gaussian distribution-based data augmentation for the quality label maps, which were described in subsection 4.3.2, in order to increase the grasp labels between the left and right contact points and reduce the uncertainty of the trained CNN, which avoids predicting the positions of the positive grasps on the edge of the object. Firstly, the positions of grasp contact points $\boldsymbol{c}^{\text{left}}$, $\boldsymbol{c}^{\text{right}}$ were estimated for each grasp. In addition, the position $\boldsymbol{g}^{\text{center}}$ and grasp quality $\boldsymbol{q}^{\text{grasp}}$ of the grasp center were given. Then, we presented a Gaussian distribution-based function f_j^{gaussian} based upon the position of contact points $\boldsymbol{c}_j^{\text{left}}$, $\boldsymbol{c}_j^{\text{right}}$ and the grasp quality w_j of the grasp $\boldsymbol{v}_j$. The pixels between two contact points were uniformly mapped to the standard Gaussian distribution function and all the possibilities in each pixel were multiplied with the quality of the corresponding grasp center. Specifically, the Gaussian distribution function can be described with the following formulation:

$$f_j^{\text{gaussian}} = f(\boldsymbol{g}_j^{\text{center}}, \boldsymbol{q}_j^{\text{grasp}}, \boldsymbol{c}_j^{\text{left}}, \boldsymbol{c}_j^{\text{right}}) \tag{4.3}$$

Then, using the Gaussian distribution-based function, as shown in Figure 4.8, the grasp central points were labeled with high grasp qualities and the contact points were treated as grasp with very small grasp qualities. Following the data augmentation based upon the gaussian distribution function f_j^{gaussian}, we estimated the qualities of all the pixels between the contact point pairs firstly and then classified all the pixels with quality possibilities into two classes (positive and negative grasp). The positive grasps means the grasp positions with higher grasp qualities than the threshold. Otherwise, the pixels with lower grasp qualities are the negative grasps. Hence, the final label maps were generated similar to the right of Fig. 4.8. Furthermore, the mask of the scene was considered during the data augmentation. Because the grasp samples can not consist of pixels with all positions along the y-axis of image coordinate system, the region of the object without data augmentation was filled with RGB value of [0, 0, 0] in black, which has no influence on the value of the loss function during the CNN training. The black with the RGB

value [0, 0, 0] has the value of 0 in the loss function.
Following this, a comparison of original and the Gaussian distribution-based augmented label maps are shown in Fig. 4.9. Specifically, the Gaussian distribution-based augmented label map had more labels than the original label map. In addition, we trained different datasets with and without the Gaussian distribution-based data augmentation and the comparison experiments were executed in order to prove the improvement of the Gaussian distribution-based data augmentation method.

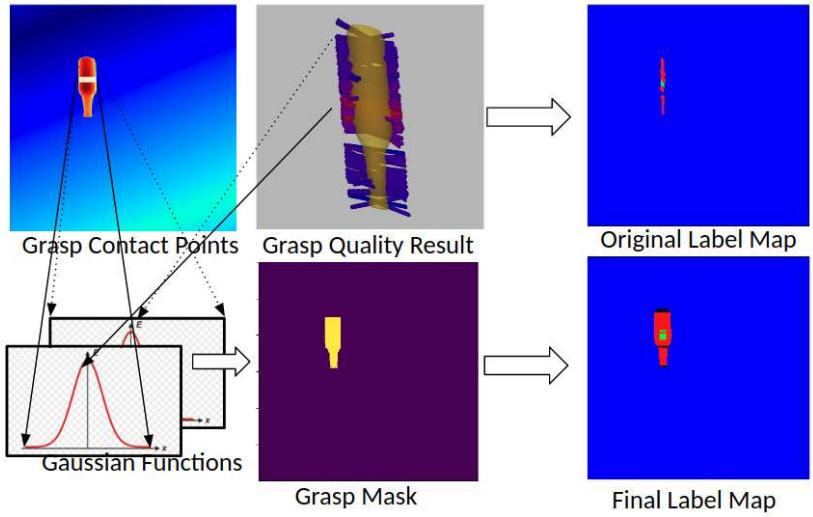


Figure 4.9: Comparison of original label map and the label maps based upon the Gaussian distribution-based data augmentation.

4.3.4 Implementation Details of the Grasp Data Generator

The related implementation details in the scenario generation and the data generation process are displayed in Table 4.1 and Table 4.2.

Table 4.1: Overview of implementation details in scenario generation.

Name	Selected	Description
y_fov	$54° - 60°$	Field of view.
aspectRatio	1.0	Aspect ratio in x and y dimensions.
elev	$0.1° - 3.0°$	Angle between z_world and z_camera.
az	$0° - 360°$	Rotation on z-axis of world coordinate.
roll	$-0.2° - 0.2°$	Rotation on camera's z-direction vector.

Table 4.2: Overview of implementation details in data generation process.

Name	Selected	Description
gripper/max_finger_width	0.085	Maximum width of gripper.
gripper/min_finger_width	0.0	Minimum width of gripper.
mesh_volume_sample	True/False	Parameter for sampling volume of mesh.
mesh_volume_max	$1e^{-3}$	Maximum volume of mesh.
mesh_volume_min	$2.7e^{-5}$	Minimum volume of mesh.
min_grasps_threshold_pre_scene	50	Minimum number of generated grasps in every scene.
default_friction_coef	0.5	Default friction coefficient for friction model.
num_cone_faces	8	Number of faces of friction polyhedral convex cone.
max_stable_pose_num	5	Maximum number of stable poses of object in table.
min_contact_dist	$8e^{-4}$	Minimum distance between two contact points.
grasp_dist_thresh	$5e^{-4}$	Threshold distance between sampling points.
metrics/quality_method	ferrari_canny	Metrics for calculating grasp quality.
metrics/soft_fingers	1	Parameter for controlling finger type in simulation.
camera/min_radius	0.65	Minimum distance between camera and table.
camera/max_radius	0.75	Maximum distance between camera and table.
camera/im_width	600	The width of synthesized image.
camera/im_height	600	The height of synthesized image.

4.4 Synthetic and Real Datasets

It should be noted that for the robotic grasping tasks, only small objects that a robotic hand could grasp were focused upon [DWL19]. The synthesized images were generated based upon our data generation

procedure and some human-labeled real images from Amazon Robotics Challenge 2017 [ZSY+18] were also considered for comparison with the synthesized images.

4.4.1 Synthetic Dataset

Our synthetic dataset considered the global features of the whole grasp scene with the camera configuration, which is different with the dataset from Dex-Net. We reduced the complexity of the grasp angle to 16 orientations. This complexity reduction makes the training process of the network more effective, since only a grasp that is parallel to the x-axis of the image has to be considered by the network. In addition, we increase the quantity of grasp samples based on the Gaussian distribution-based method, which was detailed in the last subsection. The increase in grasp quantities provides the training process with more opportunities to learn the grasp policy quickly and effectively. We stored the grasp visual model with the camera configurations; the synthetic images, the contact models containing the grasp pose, the stable pose, and others to HDF5 files as the database. In particular, the depth-jet encoded colormap served as the input data and the label maps with the Gaussian distribution-based date augmentation were separately built into the dataset. Figure 4.10 shows some examples from the synthetic dataset.

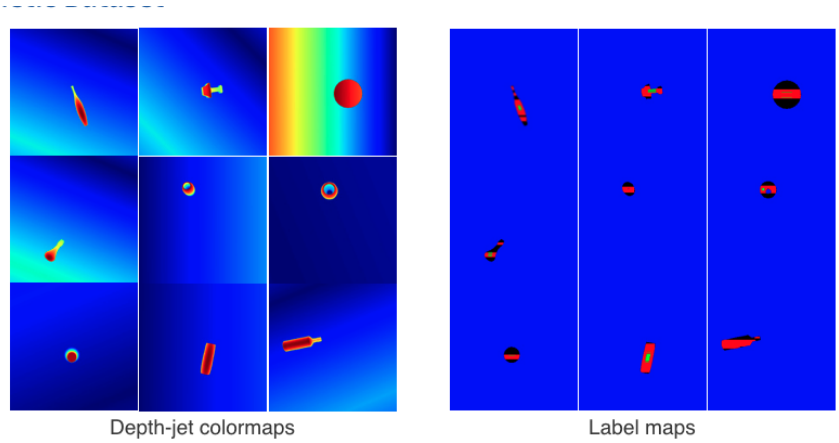


Figure 4.10: Synthetic Dataset based on the Gaussian distribution-based data augmentation. (Left) Depth-jet encoded colormaps. (Right) Labels maps based upon the Gaussian distribution-based data augmentation.

4.4.2 Real-World Dataset

In order to bridge the gap between the real image and the synthesized image, we selected one annotated real-world dataset from Amazon Robotics Challenge 2017, which contained 389 scenes and 6.2k samples [ZSY+18]. We generated an encoded depth-jet colormap with the original depth data as part of the grasping dataset, with the aim of comparing the training results between different training datasets. The images and the corresponding annotated labels are shown in Fig. 4.11.

4.4.3 Dataset List

Finally, we generated multiple versions of the datasets:
The Syn-Train: We built a synthetic training set with over 65k samples, generated according to the method described in Section 4.3 and with the Gaussian distribution-based data augmentation.
The Syn-Real-Train: At training set with 65k synthetic images with the Gaussian distribution-based data augmentation method and 6.2k real-world samples from Amazon Robotics Challenge.
The Real-Train: A training set of 6.2k real-world samples from Amazon Robotics Challenge (without the Gaussian distribution-based data augmentation).

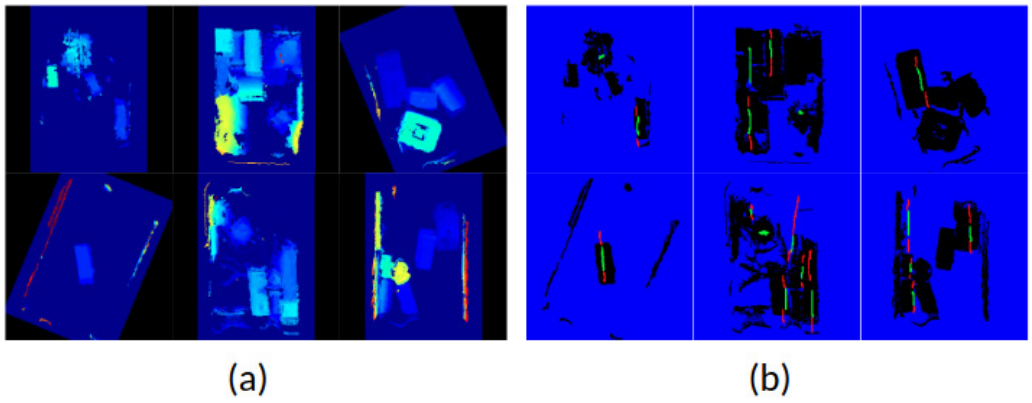


Figure 4.11: Real-world image dataset. (a) Encoded real-world images. (b) Corresponding annotations. The positive labels appear in green and the negative labels are depicted as red points.

4.5 Encoder-Decoder Network Architecture

Transfer learning is a popular method used in the network training. Convolutional neural networks pre-trained on ImageNet dataset [DDS+09] are the backbone of many state-of-the-art approaches. The knowledge of pre-trained model can be utilized in quickly training a CNN based upon a new image dataset. We produced a deep residual network to predict the possibilities of horizontal robotic grasps using depth data. The general network structure used for this experiment is basically divided into two parts: encoder and decoder. The encoder-decoder network architectures were broadly utilized in the field if object segmentation. The encoder in our network was identical to layers of ResNet-50 v2 network [HZRS16b]. Specifically, we removed the fully connected layers in the end of ResNet-50 v2 [HZRS16b] in order to keep features from input data. We trained the ResNet network using the transfer learning method based upon the ImageNet pre-trained model due to its effectiveness [DDS+09]. The decoder part consists of convolutional, batch normalization and bilinear upsampling layers, which extract the features from the results of the encoder. The output of this network is essentially different to other

architectures, which address pixel-based robotic grasp detection. The whole network structure can be found in Figure 4.12, while the individual operators are described in Table 4.3.
When building the network, we selected the ResNet-50 v2 [HZRS16b] without the pooling layer in the end and the last convolutional layer as the encoder, which make our network smaller and easier to train similar to other encoder-decoder network [BKC17]. Different activation functions in ResNet were investigated during training process, such as ReLU and tanh activation functions. Meanwhile, multiple convolutional layers, batch normalizations and bilinear upsampling layers were added to the backbone of ResNet-50 v2 with ImageNet pre-trained model and these layers built the decoder of our encoder-decoder network. The input layer size fitted the image size of 300×300 pixels with three channels, which accepted the encoded depth-jet colormap based on the depth data. The three channels of output data represented the possibility of three classes, such as the negative grasp central points in red with RGB value of [1, 0, 0], positive central points in green with RGB value of [0, 1, 0] and background in blue with RGB value of [0, 0, 1]. Both additional convolutional layers have a kernel size of 3×3 with a stride of 1×1. Batch normalization with tanh activation functions and bilinear upsampling were utilized after each additional convolutional layer. The resulting of the final layer, named the grasp affordance map or grasp inference map, has a size of $300 \times 300 \times 3$, which encodes the possibility of a robotic horizontal grasp for each pixel.

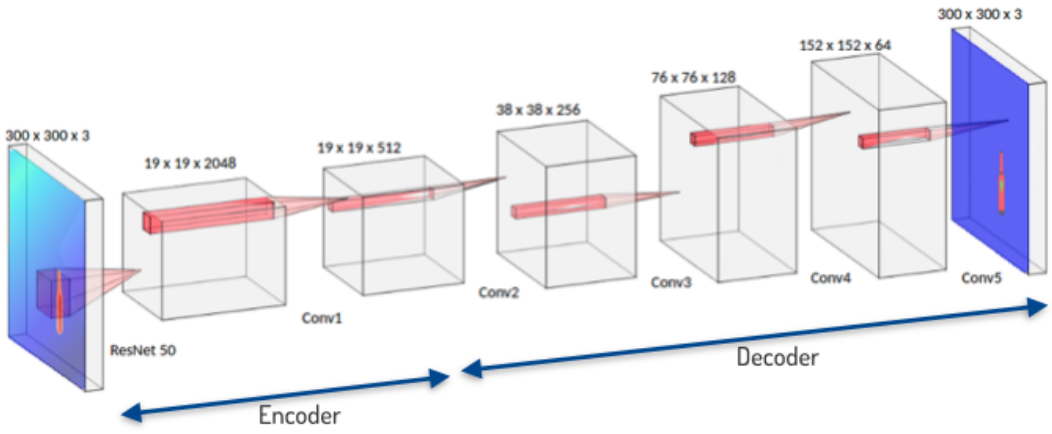


Figure 4.12: Overview of the network structure. The input of the network is represented on the left side by the encoded image and the label is shown as the affordance map on the right side.

Table 4.3: Construction of the network architecture and description of the main operations in the network.

Name	Kernel Size	Stride	Channel In/Out	Input Resolution	Output Resolution	Input
ResNet-50	-	-	3/2048	300×300	19×19	Image
conv1	3×3	1	2048/512	19×19	19×19	ResNet-50
Bilinear upsampling1	-	-	512/512	19×19	38×38	conv1
conv2	3×3	1	512/256	38×38	38×38	Bilinear upsampling1
Bilinear upsampling2	-	-	256/256	38×38	76×76	conv2
conv3	3×3	1	256/128	76×76	76×76	Bilinear upsampling2
Bilinear upsampling3	-	-	128/128	76×76	152×152	conv2
conv4	3×3	1	128/64	152×152	152×152	Bilinear upsampling3
Bilinear upsampling4	-	-	64/64	152×152	300×300	conv4
conv5	1×1	1	64/3	300×300	300×300	Bilinear upsampling4

Specifically, kernel size is the shape of the convolutions. The stride refers to the slide pixel distance of the convolutional layers. After each convolutional layer, batch normalization was applied firstly and then a tanh activation function was performed.

4.5.1 Training Procedure

We used the transfer learning method to train the grasp network with an ImageNet pre-trained model in order to transfer the domain from image recognition to image-based robotic grasping detection. Before feeding the training sample image to the network, the normalization method was utilized in the training sample for the purpose of avoiding the domain shift problem during the training procedure. When facing domain shift problem, difference of the data distributions of two dataset leads to the failures during training feedforward networks with backpropagation. The network was trained with a batch size of 32

and approximately 34.5 million weights of the network were trainable.
Specifically, the numbers of the three classes in out label maps are vastly different, which leads to a label imbalance problem. When facing the label imbalance problems during the training process, the updates of weights may fail. Due to the imbalance in the label quantities among the three classes of the training sample, a specified loss function was presented to address the problem. Specially, the difference between the numbers of the labeled grasps and the background was extremely large, which leads to the dilemma of training with a normal cross-entropy loss function. Hence, we designed the adaptable weight cross-entropy loss function as the following formulation:

$$
\begin{aligned}
L = \frac{1}{3HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{k=1}^{3} -\boldsymbol{M}_{ijk}^{\text{adaptive}} \boldsymbol{\Psi}_{ijk} \log \frac{e^{\Theta(\boldsymbol{x})_{ijk}}}{\sum_{l=1}^{3} e^{\Theta(\boldsymbol{x})_{ijl}}} \\
\boldsymbol{M}_{ijk}^{\text{adaptive}} = \sum_{k=1}^{3} \lambda_{\text{penalty}} \cdot \boldsymbol{M}_{ijk} \\
\lambda_{\text{penalty}} = \begin{cases} \frac{\sum \boldsymbol{N}^{\text{class}}}{\boldsymbol{N}_{ij}^{\text{class}}}; \boldsymbol{N}_{ij}^{\text{class}} > 0 \\ 0; \boldsymbol{N}_{ij}^{\text{class}} \leq 0 \end{cases}
\end{aligned}
\tag{4.4}
$$

where, $\Theta(\boldsymbol{x})_{ijk}$ represents the affordance map of the network and $\boldsymbol{\Psi}_{ijk} \in \mathbb{R}^{H \times W \times 3}$ denotes the ground truth tensors and $\boldsymbol{M}_{ijk}$ represents the mask for three classes: the positive grasp central point, the negative grasp central point, and the background. Due to the label imbalance problem, an adaptive penalty mask was built to adjust the weights on the pixels with different classes. In our experiments, we found that generally more penalties were placed for the negative or positive label, which improved the efficiency and stability of the training procedure. In particular, adaptive variable $\lambda_{penalty}$ will be naturally modified with regard to the label quantity distribution during network training. For training, we used 1,521 different objects in the database and generate approximately 65k training samples with more than 9.7 million labels. All the implementation details concerning the training are summarized in following Table 4.4.
In particular, we trained our network with different activation functions and optimizers. Two popular activation functions (ReLU and tanh) were investigated during the training process. While training the network with ReLU activation functions, the gradients in the framework often 0 after some global steps due to the imbalanced distribution of labels. Compared with the ReLU activation function, the tanh activation function was experimentally more suitable for our framework. In addition, we tested SGD and Adam optimizers during the training process with different learning rates and batch sizes. Ultimately, we selected the Adam optimizer for our network trained with the transfer learning method.

4.5.2 Implementation Details of the Training Procedure

Table 4.4: Overview of the implementation details during the training procedure.

Name	Setting	Description
GPU	GeForce GTX 1080Ti	GPU Hardware.
Framework	Tensorflow 1.13.1 [AAB+16]	Training inference.
Optimizer	Adam	Selected optimizer for training of network.
Learning Rate	4×10^{-2}	Selected optimal learning rate for training of our network.
Learning Rate Decay	exponential_decay	Selected learning rate decay function.
Decay Step	300000	
Decay Rate	0.77	
1st Momentum Decay β_1	0.9	Recommended value of learning rate function.
2st Momentum Decay β_1	0.999	Recommended value of learning rate function.
Loss Function Scale λ_{scale}	10	Empirically scale weight for the loss function.
Batch Size	32	Batch Size was increased according to the capacity of GPU.
Iterations	325k	Complete training global steps - each iteration contains one minibatch.
Random Noise	mean = 1, std=0.01	Random Noise is added to the input data with mean and standard deviation of the normal distribution.
Dataset	over 65k synthesized Training samples and Training with annotations	Inside the visual simulation model, there were various camera configurations and stable poses of objects taken. Each scene with enough analytical grasp labels was rendered and synthesized images were developed based on the grasps distribution.
Image Size	$300 \times 300 \times 3$	Size of encoded color map based on depth image rendered by pyrender.

4.6 Grasp Execution System with Trained Network

The grasp execution system consists of three main parts - pre-processing, the inference network, and global optimization - to achieve the final executed grasp on a real-world experiment platform.

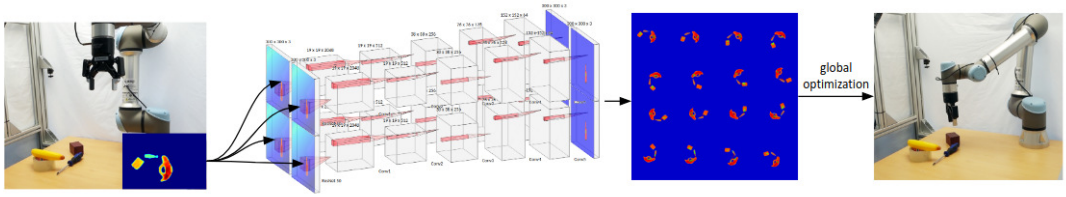


Figure 4.13: Grasp execution system with deep learning-based framework.

For the stability and proficiency of the network inference, a few basic pre-processing methods were employed in the original depth data of the 3D sensor. First, the depth image with a low resolution of 640×480 was impainted with the depth completion algorithm from [KHW18], which was used to complete the depth of sparse Lidar depth data with high speed on the CPU. During the depth completion, the holes in the depth data from the Intel Realsense Camera were filled using an estimated depth.
Following this, the inpainted depth images were encoded to the depth-jet colormap before being fed into the network. In addition, volume randomization was utilized in the data generation, which increased the diversity if the depth information. This technique increases the robustness of the network with regard to depth data observed from different camera heights. Due to its character, our synthetic dataset and the trained network were able to work on a different experimental platform, with the camera accommodated at various heights. Moreover, the depth-jet encoded color image was rotated 16 times and fed into the trained network because the design of our network was only to predict the grasps parallel to the planar table surface and the x-axis of image coordinate system. Following this, the network need not predict grasp orientation. Figure 4.13 shows the 16 grasp affordance maps predicted by the network, which identifies the horizontal grasp success possibility for each pixel in 16 rotated input images.
Following this, we designed a global optimization algorithm with a force-closure grasp filter and considered the gripper model and its parameters. Algorithm 3 introduces the whole process of the global optimization in our robotic grasping framework.
Using the global algorithm, the grasp with the higher possibility, calculated with the prediction values of the positive and negative grasp classes in the affordance maps, was selected as the candidates for the final grasp. If the grasp was executed without the global optimization method, the grasp candidate with the highest possibility was selected as the final executed grasp. These metrics for selecting grasping candidates are statistically discussed in Sec. 5.2.1. In addition, the framework without the global optimization selected the candidate with the highest possibility of a positive grasp as the final executed grasp. At the same time, in Sec. 5.2.1 compare the difference of the robotic grasping framework with and without the global optimization.

Algorithm 3: Global Optimization

Given grasp candidates $g^{\text{candidate}}$ with grasp central position $(x_{\text{camera}}^{\text{candidate}}, y_{\text{camera}}^{\text{candidate}})$ in camera coordinate system and depth values $d_{\text{world}}^{\text{candidates}}$ of the grasp samples in world coordinate system;
The grasp direction $\theta^{\text{candidate}}$ was parallel to the x-axis of the camera coordinate system;
Estimate the normal map based upon the depth image, in which the gradients of x and y dimensions in each pixel were calculated based upon the neighborhood pixels;
for *candidate in all the grasp candidates from grasp affordance map* **do**
    Calculate the two contact points $c^{\text{left\_contact}}$ and $c^{\text{right\_contact}}$ with the grasp direction $\theta^{\text{candidate}}$ and the grasp central position $(x_{\text{camera}}^{\text{candidate}}, y_{\text{camera}}^{\text{candidate}})$;
    Calculate the distance d^{contact} between two contact points;
    while *the distance* d^{contact} *is smaller than maximal gripper width* **do**
        Estimate the normal vectors of the two contact points, such as $n^{\text{left\_contact}}$ and $n^{\text{right\_contact}}$;
        Calculate the angle β between the normal vectors and the grasp direction;
        while *the angle* β *is smaller than* $\tan^{-1}\mu$*, which means the grasp direction is in friction cone. (*μ*: friction coefficient.)* **do**
            Use the normal direction of contact point as the grasp direction of final executed grasp;
        end
    end
end
If all grasp candidates were not successfully fine-tuned, the final executed grasp was selected with the grasp with the maximal grasp qualities according to the grasp affordance map. In addition, the grasp direction of the executed grasp was parallel to the x-axis of the image coordinate system.

Figure 4.14 shows the whole pipeline of global optimization. Firstly, the gradients of pixels in depth image were calculated, with the aim of finding the edge pixels in the scene. Then, the global optimization algorithm estimated two contact points for each candidate based upon the grasp central point and grasp direction, which was parallel to the x-axis of the image coordinate system, as planned in grasp affordance map. In addition, a filter of grasp width was utilized after obtaining the two contact points. The distance between contact points was calculated and the grasp candidate was considered as a valid candidate, whose distance was smaller than the maximal gripper width. Meanwhile, the width between contact points was defined as the width of the grasp. Finally, the valid candidate was adjusted based upon the antipodal or force-closure grasp metric. The angle between the grasp direction of grasp affordance map and the normal direction of contact points was compared with $\tan^{-1}\mu$, which μ means the friction coefficient. When the condition is satisfied, the grasp candidate is in the friction cone and it is an antipodal grasp. In order to improve the success rate of grasping and obtain an optimal grasp direction, the normal direction of contact points was set as the direction of the final executed grasp, which was

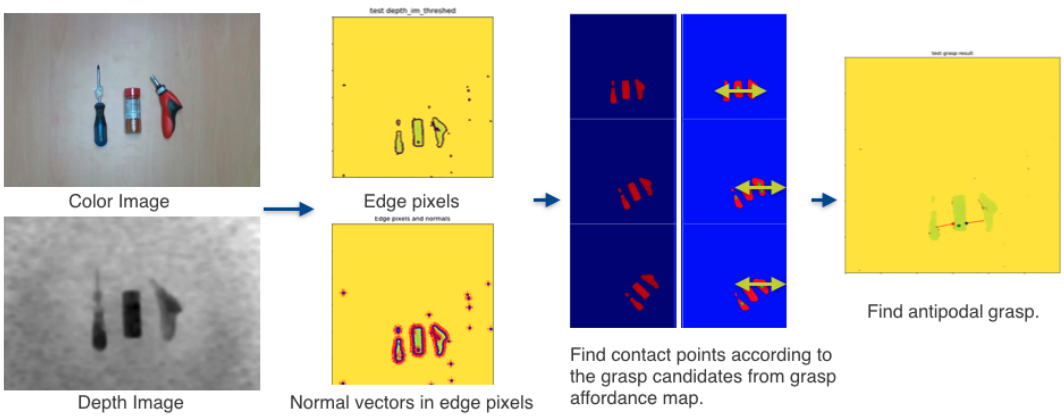


Figure 4.14: Pipeline of the global optimization for finding optimal grasp candidate.

called as grasp direction optimizer, as depicted in Figure 4.15. We utilized the filter for gripper width and the optimizer for optimal grasp direction in the global optimization algorithm to increase the robustness of the robotic grasping framework and reduce the failures caused by external factors, such as collisions during grasping. The real-world robotic experiment was tested in different experimental settings. The qualitative study and experiment results are described in Chapter 5.

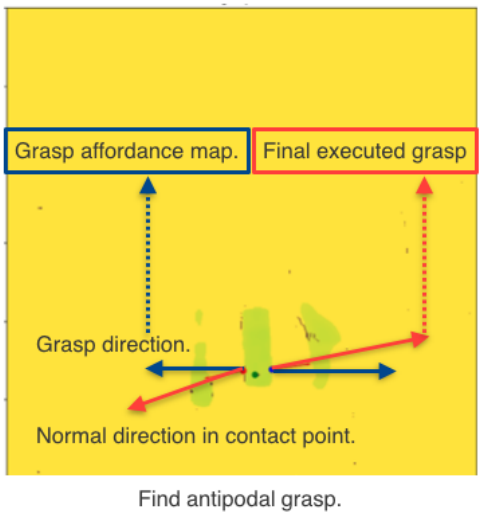


Figure 4.15: Optimal executed grasp candidate based upon the global optimization.

4.7 Conclusion

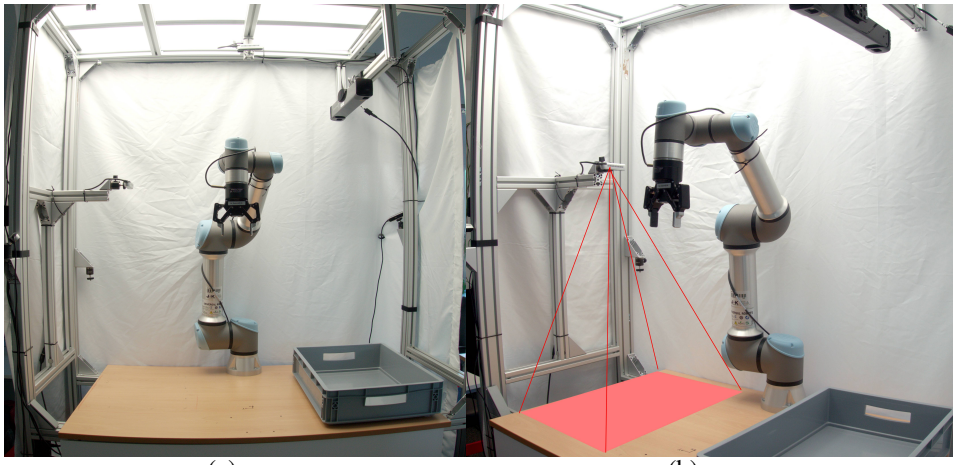
In this section, the whole pipelines of a novel data generation method for robotic grasping based on force-closure and grasp measure methods and network training, as well as a global optimization method for finding optimal grasp candidate were introduced in detail.
We found that the model trained with our generated dataset was able to avoid the lack of labels problem and predict the affordance map with a low degree of uncertainty. Because of the Gaussian distribution-based data augmentation method, the labels between the contact points were estimated based upon the grasp qualities in grasp central points and the encoder-decoder network learned the data distribution from our synthetic dataset. The dataset with the data augmentation contains more labels than the human-annotated dataset without data augmentation. Hence, the uncertainty of the trained network could be effectively reduced in comparison to a network trained with a low number of labels. When training dataset without the Gaussian distribution-based data augmentation, the possibilities of negative and positive grasps in the unlabeled pixels are not able to clarify the class which the pixels belong to. Therefore, the network model trained based upon the data augmentation need not face the problem of unpredictability in unlabeled pixels, which plays a critical role in real-world robots. Otherwise, the encoder-decoder network trained on our synthetic dataset observes global visual features of camera images, which contain the geometric information of grasped objects. Equally, our model was able to predict a robust grasp affordance map for the pixels based upon the information critical for grasping. It is for this reason that our model also performed grasping well even in a complicated scene with multiple objects. In addition, we designed a global optimization method combining the antipodal grasp method and basic image processing method together, in order to improve the robustness of the whole framework.

5 Experiments, Evaluation and Results

We tested the proposed grasping system by executing a series of experiments with a physical robot system and comparative experiments using deep learning and inference for robotic grasping with 1) a synthesized training dataset only, 2) a synthesized and annotated training dataset, 3) grasp execution without global optimization and 4) grasp execution with the global optimization method. In this chapter, details of all the experiments, the results, and the evaluation of our robotic grasping framework are presented.

5.1 Experimental Setup

We chose the Intel RealSense Depth Camera D415 as the sensor to observe the depth image of the scene with a resolution of 640×480. The physical robotic experiments were executed with a UR5e robot arm mounted with a Robotiq 2-Finger-Gripper 85. The camera was placed using an eye-to-hand configuration, as shown in Figure 5.1 (b).



(a) (b)

Figure 5.1: Experimental setup. (a) front view, (b) side view.

Furthermore, for the robotic experiments, we select 10 particular known objects from the YCB dataset [CSB+17] and 10 novel household items, as shown in Figure 5.2. Additional details of the test dataset are presented in the appendix.

(a) (b)

Figure 5.2: The testing set of 20 objects pertaining to daily life that were used for the experiments with different geometric features (spherical-like shapes, cuboid-like shapes, cup-like shapes, and complicated shapes). (a) 10 known objects in the YCB dataset, (b) 10 unknown household objects.

During the real-world robotic grasping experiments, the single object from the YCB objects and the household objects was placed on the planar table in our robot platform. Each object was to be grasped with the UR5e robot. If the grasp is failed, two lifting operations were able to move the object to be grasped. If the object was successfully grasped after lifting operation. Then, the next object selected from the test set was to be grasped. Meanwhile, three failures were calculated in the success rate. This process was repeated three times for each object in the test set and the final success rate of grasping was calculated by dividing the number of success grasps by the number of all the executed grasps. In addition, the multi-object grasping was also tested and a clean grasp task were executed during the evaluation. We placed four to six objects randomly selected from the test set on the table and the robot grasped the object one by one.

5.2 Qualitative Study

5.2.1 Statistical Horizontal Grasping Analysis

In this subsection, the affordance map of the horizontal grasps is analyzed, with the aim of understanding the distribution of the graspable and ungraspable possibilities.
Firstly, the depth data observed by sensor is encoded based upon the depth-jet encoding method, which is a commonly used method in image processing. Then, the encoded colormap is fed into the inference model with trained weights and the grasp affordance map is calculated. The inference regards the encoded depth image (the depth-jet encoded colormap) as input while the different colors of encoded depth images represent various depth values.

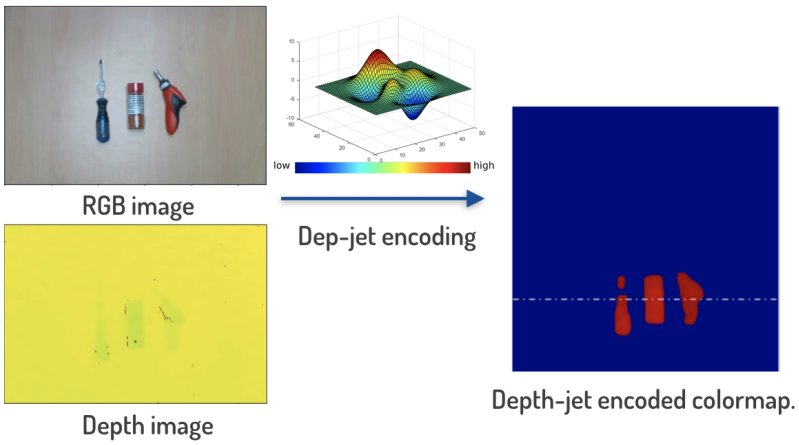


Figure 5.3: Example of the depth-jet encoded colormap.

Fig. 5.4 depicts the input for inference and the affordance maps for the horizontal grasps based upon the Syn-Train dataset with Gaussian distribution-based data augmentation, the Real-Train dataset without the data augmentation, and the Syn-Real-Train dataset, which were described in the preceding chapter. Respectively, red, green, and blue in the affordance map represented three categories: negative grasp central position with low grasp quality, positive grasp central position with high grasp quality, and background. The statistics of the affordance map in the marked position presenting grasping probability are plotted in Figure 5.4 (Right). The red curve depicts the probability curve of the negative grasp and the green curve indicates the probability distribution of the positive grasp based on the original affordance map.

Observing the networks trained with the datasets that were generated with and without the Gaussian distribution-based augmentation method, the results demonstrate that the dataset with Gaussian distribution-based augmentation performed well both from the perspective of the affordance map and the grasp probability distribution. The CNN trained based upon dataset without the data augmentation has a bad inference result in the region in which the grasp classes were not labeled and the final executed grasp may locate in the edge of object, which is negative to the grasping in a real robot. This leads to the uncertainty of the CNN trained based upon the dataset without the data augmentation. Our Gaussian distribution-based augmentation method could reduce the uncertainty of trained network and accurately and rapidly increase the labels using the statistical method.

According to the affordance map and its horizontal grasp probability distribution curves (Fig. 5.4), a grasp with horizontal orientation and a center position in the middle point of each object achieves the highest probability within the entire selected locations. The results prove that the trained network could successfully detect horizontal force-closure grasps. Furthermore, we generated a new grasp probability map by dividing the positive grasp probability by the negative grasp probability (based upon the affordance map predicted by the network trained with our Gaussian distribution-based data augmentation).

The comparison curves are shown in Figure 5.5. It is evident that the best horizontal grasp in all the grasp samples could more efficiently be selected in accordance with the second probability curve in Figure 5.5.

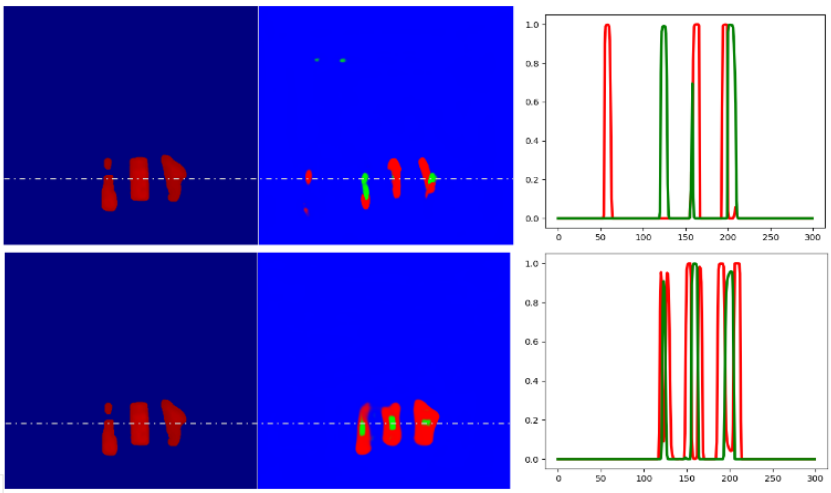


Figure 5.4: Statistical horizontal grasping analysis without and with Gaussian distribution-based data augmentation. (Left) Encoded depth-jet image, (Middle) Horizontally grasp affordance map, (Right) Grasp probability distribution in marked position (red: negative grasp possibility, green: positive grasp possibility). (Top) Trained with the Real-Train dataset without Gaussian distribution-based Augmentation. (Bottom) Trained with the Syn-Train dataset with Gaussian distribution-based augmentation.

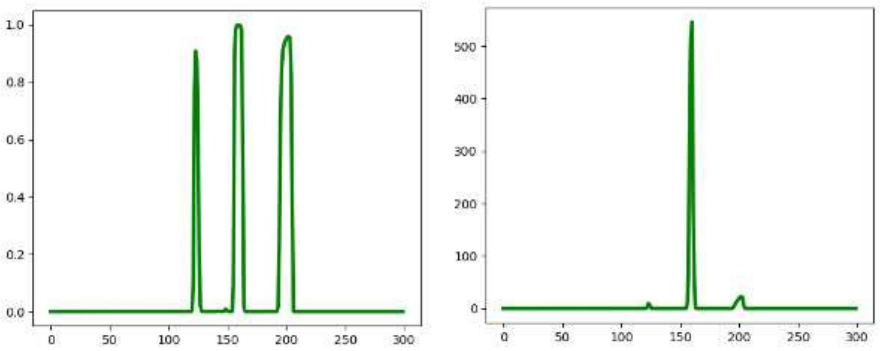


Figure 5.5: Investigation of horizontal grasp probability based on affordance map. (Left) Original positive grasp probability curve. (Right) Grasp probability curve generated with positive and negative grasp probabilities.

5.2.2 Grasping Pose Analysis

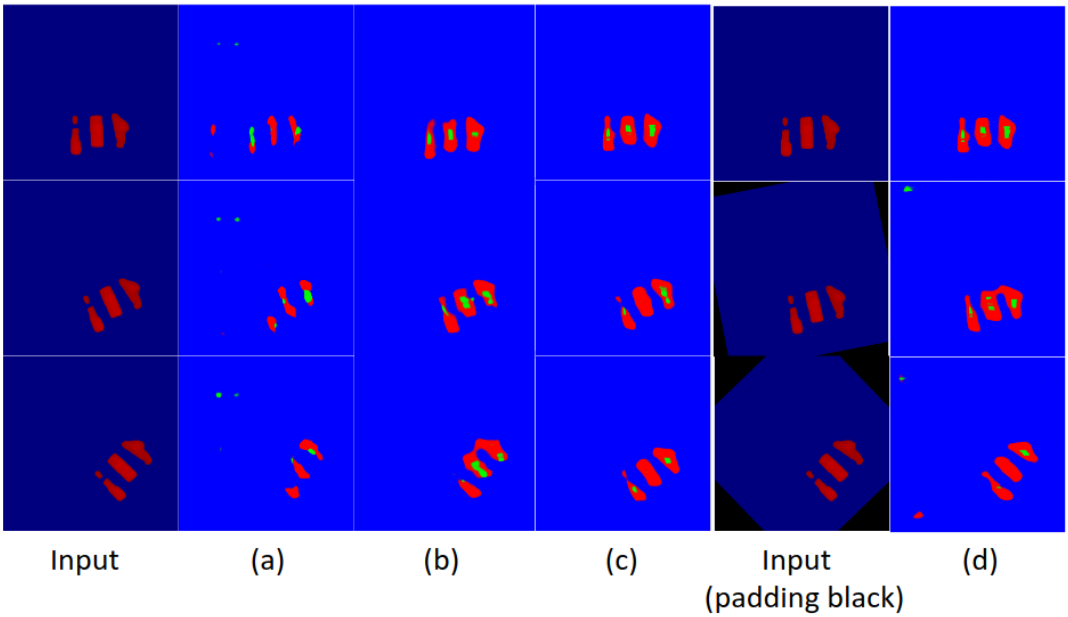


Figure 5.6: Grasping pose analysis. The rotated encoded input image was fed into a trained network and affordance maps were respectively estimated. Red: positive grasp central point, Green: negative grasp central point, Blue: background. (a) Results from a dataset without Gaussian distribution-based data augmentation. (b) Results from the Syn-Train dataset (the only synthetic training set). (c) Results from the Syn-Real-Train dataset (synthetic and real training set). Different padding colors during the rotation operations were compared between (c) and (d) with the same training set. (a) (c): padding with adaptive color, (d): black padding.

Fig. 5.6 compares the grasp affordance maps between the different training sets and various padding colors during the rotation operations before feeding input into the network. We trained the network with the Syn-Real dataset with only real data and without Gaussian distribution-based augmentation (Fig. 5.6 (a)), with a Syn-Train dataset (only synthetic data) with the Gaussian distribution-based data augmentation (Fig. 5.6 (b)), and with a Syn-Real-Train dataset (synthetic and real data) with the Gaussian distribution-based data augmentation (Fig. 5.6 (c)).

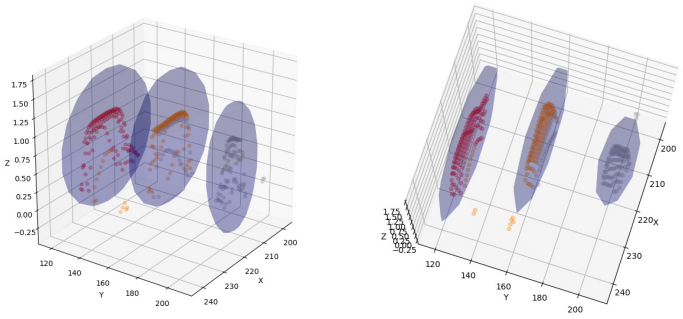
Examining the results of the network trained with the dataset (without Gaussian distribution-based augmentation), some pixels in the table were misclassified and the edge pixels even predicted positive grasps, this kind of uncertainty has a significant influence on real-world robotic grasping. Therefore, the dataset without the augmentation was not considered during the robotic experiments.

Fig. 5.6 (b) and (c) show the inference results for networks trained with only synthetic data (Syn-Train dataset) and the Syn-Real-Train dataset, containing synthetic data and little real data. The results prove

that our Syn-Train dataset with over 65k of synthetic data performs well in a scene with multiple objects. In addition, the network with the Syn-Real-Train dataset improved the grasp orientation results during rotations of the input image.
We also presented an adaptive depth-based padding method for the rotation operation and investigated the results of input images with adaptive padding color and black color, used by [CCZS19] and [ZSY+18]. From Fig. 5.6 (d), we observed that a few pixels on the table were predicted in green and red. This will reduce the stability and ability of the entire framework.

5.2.3 Investigation of the Grasp Quality Affordance Map with the Gaussian Mixture Model

For the purpose of investigating whether our Gaussian distribution-based method did reduce the uncertainty in the trained network, we used the encoded depth image and predictions of the grasp quality and mask to examine whether our inference results could be presented with Gaussian Mixture Model (GMM), which is frequently used in the statistical analysis of data. Fig. 5.7 shows the GMM-based optimization result of the affordance map from Fig. 5.4 (Bottom Middle). The XY coordinate system represents the image coordinate system and the Z coordinate is related to the grasp quality for each pixel. Specifically, in order to accelerate the computation, the affordance map for a positive grasp (the second channel of the prediction result) was clipped to 0.09-1.00 and the pixel in the affordance map was considered if the possibility was more than 0.1. According to the result in Fig. 5.7, the grasp probability distribution for clutter scenarios can be described with GMM. Taking into consideration of the geometric information and the grasp affordance map, the grasp affordance map can be adapted with GMMs and divided into multiple clusters, which represent different distributions of the grasp probabilities. Additionally, the best grasp poses for each object in the scene were located in the center of each Gaussian component. This proves that the inference result meets the GMM without noise in the pixels of the table and the edges of objects.



(a) (b)

Figure 5.7: GMM-based grasp optimization from different viewpoints.

5.3 Evaluation Details

We evaluated the capability of our grasp network to grasp an individual object in a cluttered scene on the real-world robotic grasping system. We investigated our grasp system in various scenarios by adjusting training datasets and system pipeline. These implementations are detailed in the following subsections. Finally, the evaluation metrics for the experiments are defined.

5.3.1 Dataset Evaluation

We generated the following diverse robotic grasping datasets based on our data generation method with or without Gaussian distribution-based data augmentation as well as with or without the small-large real-world images:
- Real-Train: 6.2k real-world images with post-processing using depth-jet encoding and without Gaussian distribution-based data augmentation.
- Syn-Train: Synthetic training set with over 65k samples as described in Sec. 4.4.1.
- Syn-Real-Train: Training set with 65k synthetic images and 6.2k real-world samples (Sec. 4.4.2).
The datasets were investigated within the real-world robotic grasping platform and the evaluation results of the grasp success rate were utilized in optimizing the data generation pipeline.

5.3.2 Execution Systems for estimating Robotic Grasping in a Real-world System

These systems consisted of the following:
- Networked-without-GO: A networked system without final global optimization based on force-closure, which directly executes grasping according to the grasp affordance map.
- Networked-with-GO: A networked system with global optimization for the purpose of obtaining the best grasp in the grasp affordance map with attention to the force-closure metric and gripper model.

5.3.3 Grasp Scenarios for Robotic Grasping Experiments

These scenarios consisted of the following: - Scenarios with Individual Objects: For each trial, one grasped object was placed on the robotic table in a random position.
- Cluttered Scenarios with Multiple Objects: Multiple objects were selected from the test set with 10 known YCB objects and 10 novel household objects and used to build different scenarios for evaluating our algorithms. - Task-oriented Scenarios with Multiple Data Lines: To demonstrate real-world applications in the field of industry and automation, we evaluated the networked system with global optimization in a scenario in which various data lines were placed on the table.

5.3.4 Evaluation Metrics

We defined the evaluation metric using the grasp success rate, which was calculated through the number of successful tests divided by the complete number of trials. As sometimes the robot will repeated the same failure grasps, the test was defined as a "failure" when the object was not successfully grasped

within two lifting operations and the reasons of failures were same. Otherwise, the failure was treated as a new failure grasp. Two objects grasped in one trial were also defined as a failure.
During the evaluation process, the final estimated grasp was executed. If the single object was not successfully grasped, one lifting operation was executed and the object was grasped again.In addition, the failed grasped object can be lifted up to three times. Finally, the grasp success rates were the evaluation metrics. In addition, the trials were repeated three times.

5.4 Results

This section details the results of the experiments based on the altered training datasets and system structures previously defined. The first manipulation test consisted of robotic grasp experiments with networks trained with different training datasets (Syn-Train, Syn-Real-Train) and diverse execution systems were also employed in evaluating robotic grasping with 3D sensors in divergent grasping scenarios. The second test was the comparative experiments conducted with the state-of-the-art published algorithms, including Dex-Net 2.0 [MLN+17] [MMS+19] and Metagrasp [CCZS19]. The algorithms were quantitatively executed in the same scene to diminish the impact of the experimental settings on the grasp success rate. The third test was a closed-loop clean grasp test in a multi-object scenario in which the robot grasped all the objects on the table. In addition, with a focus on the application of the deep learning method for robotic grasping with 3D sensors in industry and automation, the task-oriented capability of our networked execution system was investigated in an ordered scenario with multiple data lines. The results of each experiment are recorded in the following sections.

5.4.1 Real-world Grasping of Known and Novel Objects

The 20 test objects selected from the YCB dataset and everyday items were tested in our physical robotic grasping environment. The results from the 10 known YCB objects grasping test are depicted in Figure 5.8. The three variants within our method were Syn-Train+Networked-without-GO, Syn-Real-Train+Networked-without-GO, and Syn-Real-Train+Networked-with-GO. The last version of our networked system, which was trained with many synthetic and only a few real samples and assembled with global optimization based upon a force-closure grasp metric, demonstrated the best ability to grasp objects of numerous shape representations in real robotic environments. The Metagrasp pipeline also achieved a high success rate with different shapes of normal size, but was less successful with particularly small objects, such as golf balls and strawberries. This may be due to that the dataset of Metagrasp does not consider small objects. In addition, the affordance map indicates that the best grasp was achieved on the table. This may be due to the fact that Metagrasp only considers the color information of the scene. Then, data from different cameras may lead to the domain shift problem. The DexNet 2.0 also had a low success rate for small objects with spherical-like shapes. It indicated that the DexNet 2.0 had still limitation for grasping small objects with regard of the large scale dataset that they generated.
The outcomes for unknown objects selected from everyday items are shown in Figure 5.9. From these results, we can confirm that our methods are able to evaluate robotic grasping tasks for unknown objects.

A network trained with our synthetic dataset could work well with real-world data. In addition, our methods using global optimization demonstrated the base capability to grasp objects with complex structures. The Dex-Net 2.0 demonstrated a limited performance for objects with multiple local structures, perhaps indicating the constraint of only using local feature-based grasp representation from the point of view of a real-world application.

Table 5.1 shows the statistical results including those for grasping 10 YCB objects, 10 household objects, and the objects in the multi-object scenario and the experimental settings and outcomes for other state-of-the-art methods with individual two-jaw grippers. The testing objects were equivalent in all the comparison experiments. The Dex-Net 2.0 and MetaGrasp algorithms were utilizing in grasping the objects from the test set. In addition, three versions were tested based upon our robotic grasping pipeline, such as Syn-Train+Networked-without-GO, Syn-Real-Train+Networked-without-GO and Syn-Real-Train+Networked-with-GO. Specifically, the first part of the name of version means the training dataset, which has been introduced. The Networked-without-GO refers to the pipeline without the final global optimization algorithm. The Syn-Real-Train+Networked-with-GO represents the robotic grasping framework with final global optimization based upon the CNN trained with the Syn-Real-Train dataset, which consists of over 65k synthetic data and 6.2k real-world data. It is evident that the pipeline with the best performance was the third version of our method (Syn-Real-Train+Networked-with-Go), which indicates that our deep learning-based data-driven method with global optimization is beneficial for unknown object robotic grasping tasks from the point of view of research and application.

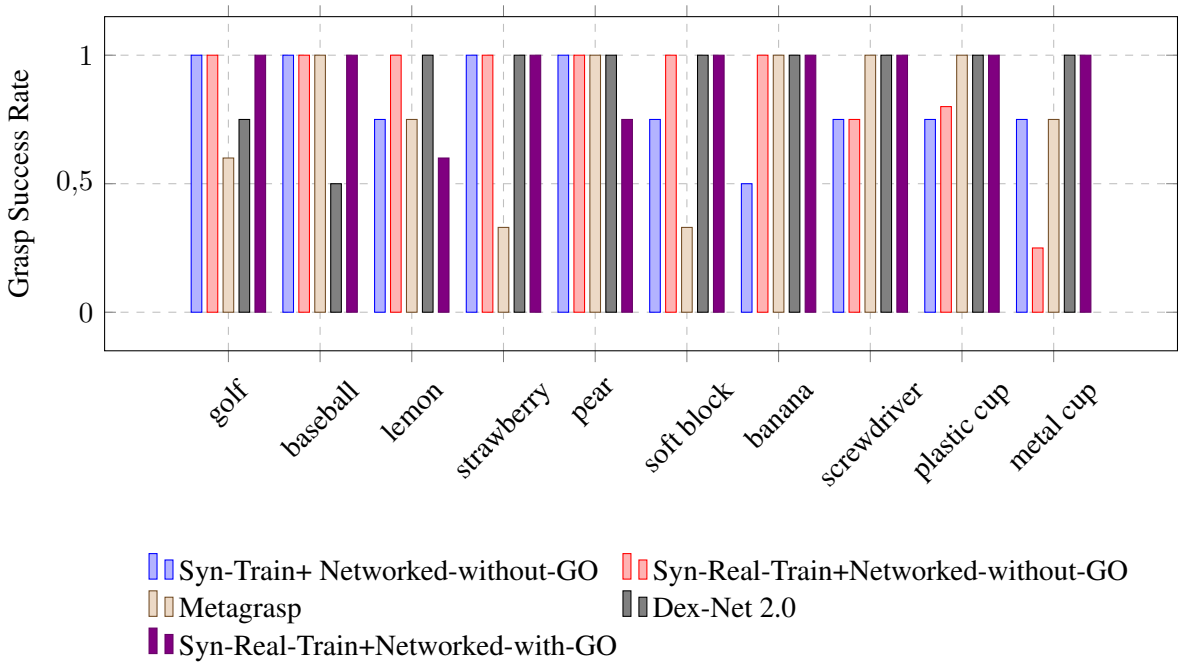


Figure 5.8: Success rate for physical grasping over YCB objects.

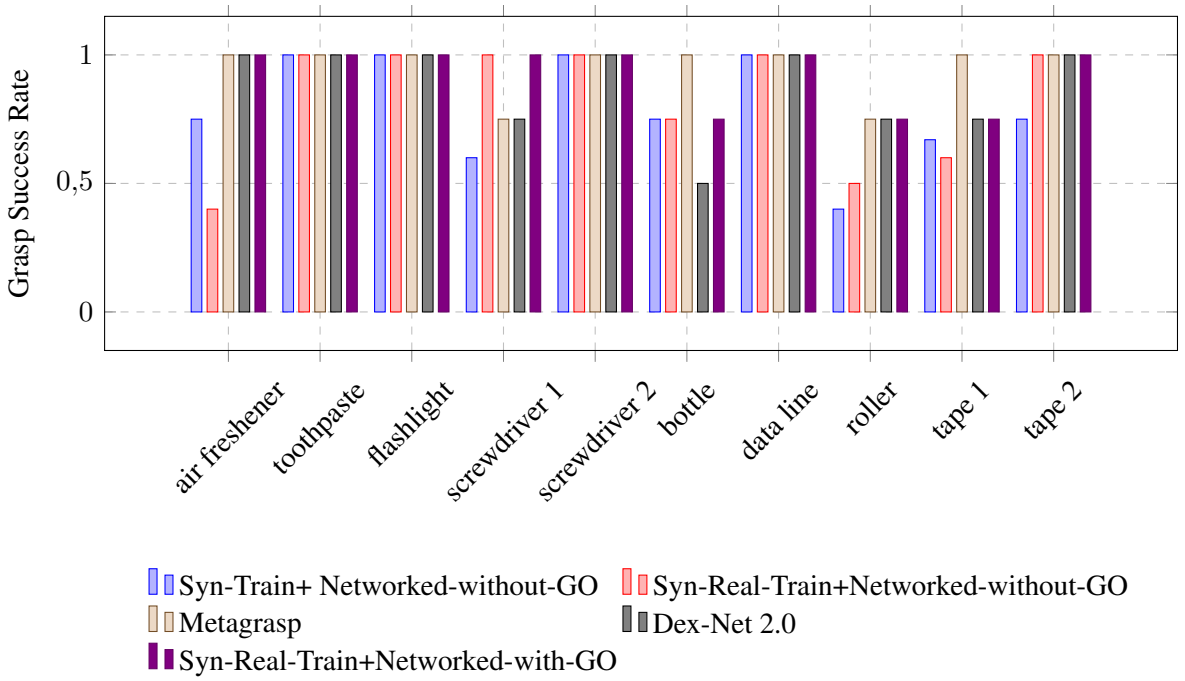


Figure 5.9: Success rate for physical grasping over household objects.

Table 5.1: Outcomes of grasping comparison experiments

Method	Settings		Success Rate (%)		
	Objects	Trials	YCB-obj	household-obj	multi-obj
DexNet 2.0*	20	145	88.24	83.33	80.00
Metagrasp*	20	148	67.44	**93.75**	79.45
Syn-Train+ Networked-without-GO*	20	174	80.56	75.00	60.20
Syn-Real-Train+ Networked-without-GO*	20	165	85.29	75.00	62.64
Syn-Real-Train+ Networked-with-GO*	20	136	**90.91**	90.91	**85.71**

* Success rates were based upon testing using a UR5e physical robot. The 10 known objects were selected from the YCB dataset and the novel testing set was 10 household objects.

5.4.2 Task-oriented Grasping by Physical Robot

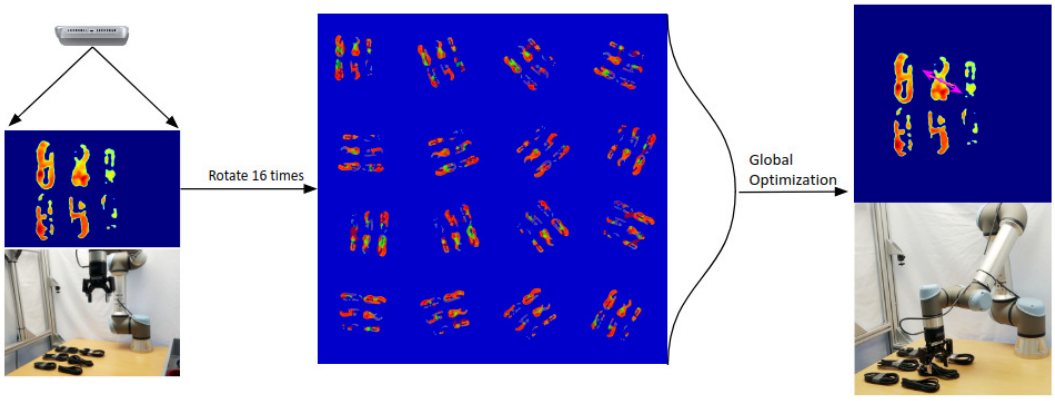


Figure 5.10: The pipeline of task-oriented robotic grasping experiment with our grasping framework. Dataflow: Depth-jet encoded colormap, affordance map, final grasp.

It is recognized that the robotic grasping of objects of complex objects is a difficult task in research and industry. In this experiment, we tested our robotic grasping framework in a scenario in which we positioned multiple data lines that had a complicated structure. Fig. 5.10 shows the main pipeline of task-oriented real-world robotic grasping. We tested for the clean grasping of data lines twice and grasped 12 data lines in a scenario with 13 grasps with a success rate of 92.31%. The result suggests that our robotic grasping method is a robust and effective solution for unknown object grasping in complicated scenarios.

6 Conclusion

This thesis leverages recent advances in deep learning to present an innovative method of data generation based upon Gaussian distribution-based data augmentation and robust analytical methods to construct a grasping model and build a novel robust robotic grasping framework with a low-resolution 3D-sensor. Through the evaluation of quantitative studies and real-world robotic experiments, our framework achieved a state-of-the-art performance for known and unknown objects robotic grasping in complex scenarios and for a challenging specialized task-oriented activity in industry in comparison to some published methods.

Initially, in order to build a robust and reliable dataset for our robotic grasping task, we proposed a new data generation method for this kind of robotic grasping. At the same time, a novel Gaussian distribution-based data augmentation for grasping data generation was presented, which plays an important role in the high robustness and the performance of the designed neural network. Following this, our data generation pipeline utilized the Gaussian distribution-based data augmentation method and multi-threading to efficiently generate a large number of robotic grasp samples with reliable labels. Specifically, the labels were evaluated with the force-closure and grasp measure metrics, which improved the grasp dataset reliability using analytical methods. The trained neural network with our generated datasets obtained excellent results, and the uncertainty present in robotic grasping based on deep learning methods was able to be suppressed.

Meanwhile, with the purpose of generating a robotic grasping dataset suitable for experimental setups with different camera heights, we creatively utilized the depth-jet encoding method with the depth data. The object volume randomization during the data generation process and the encoding method allowed the trained network to work stably on different experimental platforms.

Based on the data generation pipeline, we generated the Syn-Train dataset with over 65k of synthetic images, which were generated based upon 1,512 object meshes. The meshes database contained various types of shapes, including spherical-like shapes, cuboid-like shapes, cup-like shapes, complicated shapes. Similarly, we processed 6.2k of real-world encoded images with a few human annotations based on the depth-jet encoding method as part of our training set.

Following this, the transfer learning approach was applied to train the designed CNN based on ResNet-50 v2. The network with the ImageNet-based pre-trained model for object recognition tasks was transferred to the image-based robotic grasping task with the synthetic datasets. Then, the designed network was trained with the synthetic datasets with a large amount of data and limited human-annotated data from Amazon Robotics Challenge 2017. We also presented an adaptive loss function for a robust training with imbalanced labels.

In addition, our grasping framework combined the affordance map predicted by the CNN and the global optimization method, with consideration given to the real-world gripper model and force-closure metric,

to increase the robustness of deep learning-based robotic grasping with the statistical method. The candidates estimated by the affordance map were filtered by means of global optimization and the final grasp pose was executed on a real-world experiment platform.

Following this, we conducted a qualitative analysis of the robotic grasping framework using different parameters during the data generation process, the training procedure and the framework structures. This analysis confirmed that our methods significantly improved the performance of the robotic grasping pipeline.

Finally, to evaluate our deep learning-based robotic grasping framework, we built a real-world experiment platform with a UR5e robot, a Robotiq 2-Finger-Gripper, and an Intel RealSense Depth Camera D415. The comparative experiments were executed in the environment. Then, we selected 10 known objects from the YCB dataset and 10 unknown commonly used household objects as a test set. The real-world robotic grasp experiments with diverse scenarios were executed using our method and different deep learning-based published techniques. In the physical experiments, the robot using our approach was able to successfully grasp known or unknown individual objects with different shapes with a 90.91% success rate and grasp objects in multi-object scenarios with an 85.71% overall success rate. These rated offer a significant improvement when compared with the rates of Dex-Net 2.0 and Metagrasp. Our pipeline was also tested in a challenging task-oriented robotic grasping environment, in which there were several ordered data lines on the table and our pipeline was able to grasp the data lines well with a 92.31% success rate.

In summary, we have presented a new robust and high-performance deep learning-based robotic grasping framework for low-resolution 3D-sensors with a corresponding data generation pipeline combining the Gaussian distribution-based data augmentation method and robust analytical methods that trains an agile CNN using synthetic datasets with a large amount of data and a small amount of real-world data with few human annotations. We also utilized a global optimization method taking into consideration the model of real-world gripper and force-closure model of the grasp candidates from network inferences to find the final robust executed grasp.

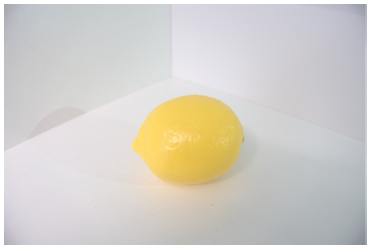
In the future, we would like to improve the data generation with more complicated scenarios and investigate the problems during transferring synthetic data of robotic grasping to real-world data of various sensors. In the meantime, we plan to utilize the 3D deep learning method to evaluate the inference framework.

A Appendix

(a) ycb banana

(b) ycb baseball

(c) ycb golf



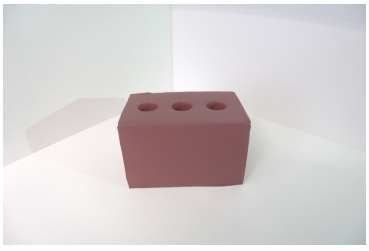
(d) ycb lemon

(e) ycb metal cup

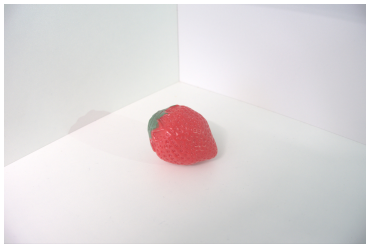
(f) ycb pear

(g) ycb plastic cup

(h) ycb screwdriver



(i) ycb soft block



(j) ycb strawberry

Figure A.1: Selected 10 known objects from YCB dataset

(a) air freshener

(b) bottle

(c) data line

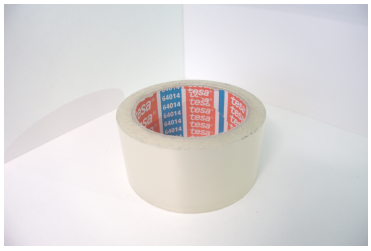
(d) flashlight

(e) roller

(f) screwdriver 1

(g) screwdriver 2

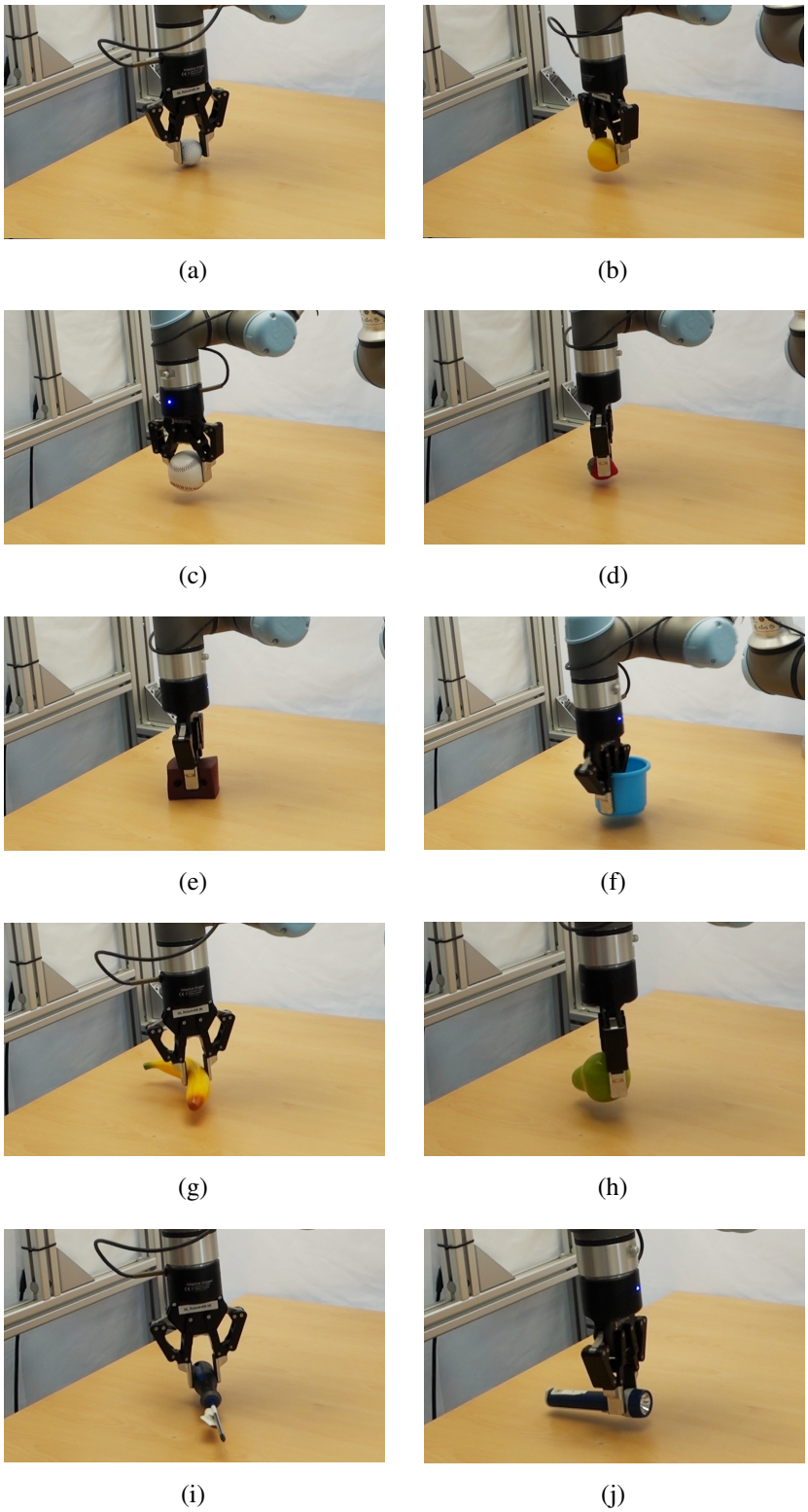
(h) tape 1



(i) tape 2

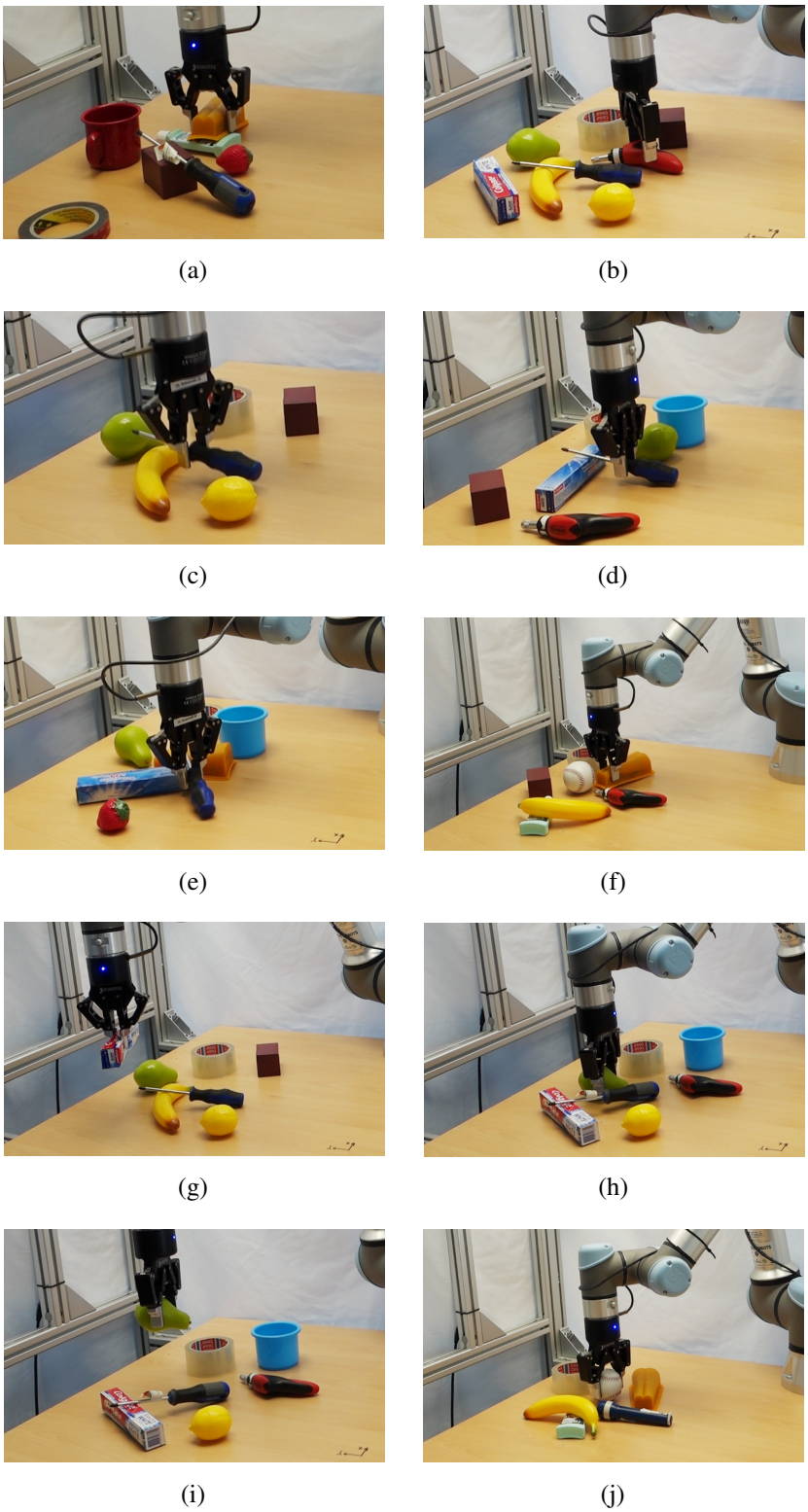
(j) toothpaste

Figure A.2: Selected 10 novel objects from daily necessities.



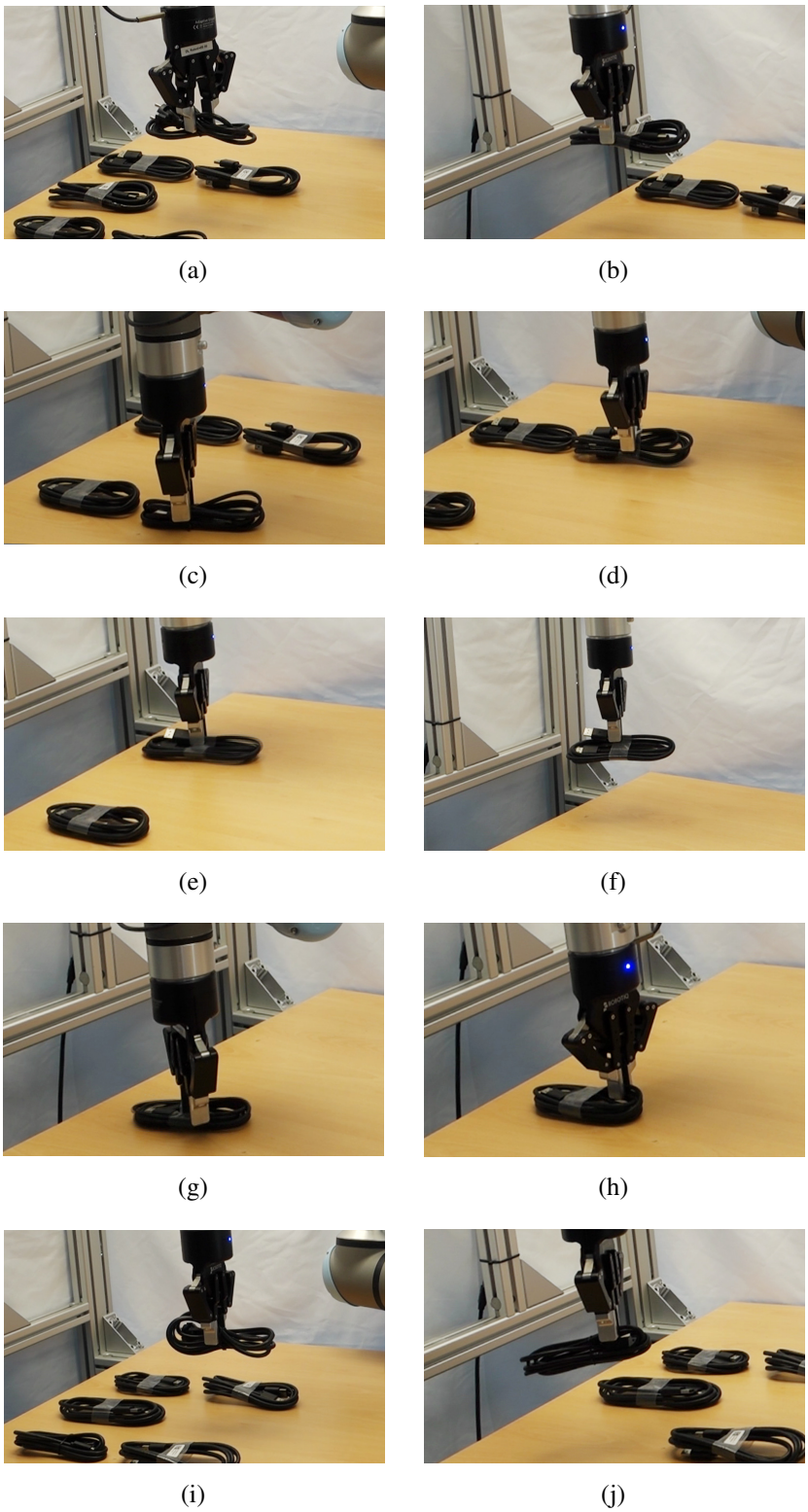
(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure A.3: Exapmles of Robotic Grasping Experiments (Single Object).



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure A.4: Exapmles of Robotic Grasping Experiments (Complex Scenarios).



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure A.5: Exapmles of Task-oriented Robotic Grasping Experiments .

Bibliography

[AAB+16] ABADI, Martín ; AGARWAL, Ashish ; BARHAM, Paul ; BREVDO, Eugene ; CHEN, Zhifeng ; CITRO, Craig ; CORRADO, Greg S. ; DAVIS, Andy ; DEAN, Jeffrey ; DEVIN, Matthieu u. a.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. In: *arXiv preprint arXiv:1603.04467* (2016)

[ATH19] ASIF, Umar ; TANG, Jianbin ; HARRER, Stefan: Densely Supervised Grasp Detector (DSGD). In: *Proceedings of the AAAI Conference on Artificial Intelligence* Bd. 33, 2019, S. 8085–8093

[ATY+19] ALOM, Md Z. ; TAHA, Tarek M. ; YAKOPCIC, Chris ; WESTBERG, Stefan ; SIDIKE, Paheding ; NASRIN, Mst S. ; HASAN, Mahmudul ; VAN ESSEN, Brian C. ; AWWAL, Abdul A. ; ASARI, Vijayan K.: A state-of-the-art survey on deep learning theory and architectures. In: *Electronics* 8 (2019), Nr. 3, S. 292

[Ben13] BENGIO, Yoshua: Deep learning of representations: Looking forward. In: *International Conference on Statistical Language and Speech Processing* Springer, 2013, S. 1–37

[BFH99] BORST, Ch ; FISCHER, Max ; HIRZINGER, Gerd: A fast and robust grasp planner for arbitrary 3D objects. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)* Bd. 3 IEEE, 1999, S. 1890–1896

[BFH04] BORST, Ch ; FISCHER, Max ; HIRZINGER, Gerd: Grasp planning: How to choose a suitable task wrench space. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004* Bd. 1 IEEE, 2004, S. 319–325

[Bic00] BICCHI, Antonio: Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity. In: *IEEE Transactions on robotics and automation* 16 (2000), Nr. 6, S. 652–662

[Bis06] BISHOP, Christopher M.: *Pattern recognition and machine learning*. springer, 2006

[BKC17] BADRINARAYANAN, Vijay ; KENDALL, Alex ; CIPOLLA, Roberto: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. In: *IEEE transactions on pattern analysis and machine intelligence* 39 (2017), Nr. 12, S. 2481–2495

[BMAK13] BOHG, Jeannette ; MORALES, Antonio ; ASFOUR, Tamim ; KRAGIC, Danica: Data-driven grasp synthesis—a survey. In: *IEEE Transactions on Robotics* 30 (2013), Nr. 2, S. 289–309

[Car12] CARBONE, Giuseppe: *Grasping in Robotics*. Springer Publishing Company, Incorporated, 2012. – ISBN 1447146638, 9781447146636

[CCZS19] CAI, Junhao ; CHENG, Hui ; ZHANG, Zhanpeng ; SU, Jingcheng: MetaGrasp: Data Efficient Grasping by Affordance Interpreter Network. In: *2019 International Conference on Robotics and Automation (ICRA)* IEEE, 2019, S. 4960–4966

[CF92] CANNY, J ; FERRARI, C: Planning optimal grasps. In: *Proc. Conf. on Robotics and Automation (ICRA)* Bd. 1992, 1992, S. 2290–2295

[CRC18] CALDERA, Shehan ; RASSAU, Alexander ; CHAI, Douglas: Review of deep learning methods in robotic grasp detection. In: *Multimodal Technologies and Interaction* 2 (2018), Nr. 3, S. 57

[CSB+17] CALLI, Berk ; SINGH, Arjun ; BRUCE, James ; WALSMAN, Aaron ; KONOLIGE, Kurt ; SRINIVASA, Siddhartha ; ABBEEL, Pieter ; DOLLAR, Aaron M.: Yale-CMU-Berkeley dataset for robotic manipulation research. In: *The International Journal of Robotics Research* 36 (2017), Nr. 3, S. 261–268

[Daw] DAWSON-HAGGERTY ET AL.: *trimesh*. https://trimsh.org/

[DDS+09] DENG, Jia ; DONG, Wei ; SOCHER, Richard ; LI, Li-Jia ; LI, Kai ; FEI-FEI, Li: Imagenet: A large-scale hierarchical image database. In: *2009 IEEE conference on computer vision and pattern recognition* Ieee, 2009, S. 248–255

[DLSX00] DING, Dan ; LIU, Yun-Hui ; SHEN, Yan-Tao ; XIANG, Guo-Liang: An efficient algorithm for computing a 3D form-closure grasp. In: *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)* Bd. 2 IEEE, 2000, S. 1223–1228

[DUB+17] DROST, Bertram ; ULRICH, Markus ; BERGMANN, Paul ; HARTINGER, Philipp ; STEGER, Carsten: Introducing mvtec itodd-a dataset for 3d object recognition in industry. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2017, S. 2200–2208

[DWL19] DU, Guoguang ; WANG, Kai ; LIAN, Shiguo: Vision-based Robotic Grasping from Object Localization, Pose Estimation, Grasp Detection to Motion Planning: A Review. In: *arXiv preprint arXiv:1905.06658* (2019)

[ESS+15] EITEL, Andreas ; SPRINGENBERG, Jost T. ; SPINELLO, Luciano ; RIEDMILLER, Martin ; BURGARD, Wolfram: Multimodal deep learning for robust RGB-D object recognition. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* IEEE, 2015, S. 681–687

[GBC16] GOODFELLOW, Ian ; BENGIO, Yoshua ; COURVILLE, Aaron: *Deep Learning*. MIT Press, 2016. – http://www.deeplearningbook.org

[HZRS16a] HE, Kaiming ; ZHANG, Xiangyu ; REN, Shaoqing ; SUN, Jian: Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, S. 770–778

[HZRS16b] HE, Kaiming ; ZHANG, Xiangyu ; REN, Shaoqing ; SUN, Jian: Identity mappings in deep residual networks. In: *European conference on computer vision* Springer, 2016, S. 630–645

[IS15] IOFFE, Sergey ; SZEGEDY, Christian: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *arXiv preprint arXiv:1502.03167* (2015)

[KB14] KINGMA, Diederik P. ; BA, Jimmy: Adam: A method for stochastic optimization. In: *arXiv preprint arXiv:1412.6980* (2014)

[KGK+18] KRAGIC, Danica ; GUSTAFSON, Joakim ; KARAOGUZ, Hakan ; JENSFELT, Patric ; KRUG, Robert: Interactive, Collaborative Robots: Challenges and Opportunities. In: *IJCAI*, 2018, S. 18–25

[KHW18] KU, Jason ; HARAKEH, Ali ; WASLANDER, Steven L.: In Defense of Classical Image Processing: Fast Depth Completion on the CPU. In: *2018 15th Conference on Computer and Robot Vision (CRV)* IEEE, 2018, S. 16–22

[KSH12] KRIZHEVSKY, Alex ; SUTSKEVER, Ilya ; HINTON, Geoffrey E.: Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*, 2012, S. 1097–1105

[LLS15] LENZ, Ian ; LEE, Honglak ; SAXENA, Ashutosh: Deep learning for detecting robotic grasps. In: *The International Journal of Robotics Research* 34 (2015), Nr. 4-5, S. 705–724

[LML+19] LIANG, Hongzhuo ; MA, Xiaojian ; LI, Shuang ; GÖRNER, Michael ; TANG, Song ; FANG, Bin ; SUN, Fuchun ; ZHANG, Jianwei: Pointnetgpd: Detecting grasp configurations from point sets. In: *2019 International Conference on Robotics and Automation (ICRA)* IEEE, 2019, S. 3629–3635

[LP17] LYNCH, K.M. ; PARK, F.C.: *Modern Robotics*. Cambridge University Press, 2017 https://books.google.de/books?id=YUkTnQAACAAJ. – ISBN 9781107156302

[LS16] LIN, Yun ; SUN, Yu: Task-oriented grasp planning based on disturbance distribution. In: *Robotics Research*. Springer, 2016, S. 577–592

[LZC+19] LIN, Huifeng ; ZHANG, Tong ; CHEN, Zhaopeng ; SONG, Haina ; YANG, Chenguang: Adaptive fuzzy Gaussian mixture models for shape approximation in Robot Grasping. In: *International Journal of Fuzzy Systems* 21 (2019), Nr. 4, S. 1026–1037

[Mat] MATHWORKS: *Jet colormap array - MATLAB jet - MathWorks Deutschland*. https://de.mathworks.com/help/matlab/ref/jet.html. https://de.mathworks.com/help/matlab/ref/jet.html

[Mat19] MATL, Matthew: *pyrender*. https://github.com/mmatl/pyrender.git, 2019

[MCL18] MORRISON, Douglas ; CORKE, Peter ; LEITNER, Jürgen: Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. In: *arXiv preprint arXiv:1804.05172* (2018)

[MEF19] MOUSAVIAN, Arsalan ; EPPNER, Clemens ; FOX, Dieter: 6-DOF GraspNet: Variational Grasp Generation for Object Manipulation. In: *arXiv preprint arXiv:1905.10520* (2019)

[MKCA03] MILLER, Andrew T. ; KNOOP, Steffen ; CHRISTENSEN, Henrik I. ; ALLEN, Peter K.: Automatic grasp planning using shape primitives. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)* Bd. 2 IEEE, 2003, S. 1824–1829

[MLN+17] MAHLER, Jeffrey ; LIANG, Jacky ; NIYAZ, Sherdil ; LASKEY, Michael ; DOAN, Richard ; LIU, Xinyu ; OJEA, Juan A. ; GOLDBERG, Ken: Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In: *arXiv preprint arXiv:1703.09312* (2017)

[MMS+19] MAHLER, Jeffrey ; MATL, Matthew ; SATISH, Vishal ; DANIELCZUK, Michael ; DEROSE, Bill ; MCKINLEY, Stephen ; GOLDBERG, Ken: Learning ambidextrous robot grasping policies. In: *Science Robotics* 4 (2019), Nr. 26, S. eaau4984

[Mur17] MURRAY, Richard M.: *A mathematical introduction to robotic manipulation*. CRC press, 2017

[Ngu87] NGUYEN, V-D: Constructing stable grasps in 3D. In: *Proceedings. 1987 IEEE International Conference on Robotics and Automation* Bd. 4 IEEE, 1987, S. 234–239

[Nie15] NIELSEN, Michael A.: *Neural Networks and Deep Learning*. 2015. – http://www.neuralnetworksanddeeplearning.com

[PBPPM09] POPESCU, Marius-Constantin ; BALAS, Valentina E. ; PERESCU-POPESCU, Liliana ; MASTORAKIS, Nikos: Multilayer perceptron and neural networks. In: *WSEAS Transactions on Circuits and Systems* 8 (2009), Nr. 7, S. 579–588

[QSMG17] QI, Charles R. ; SU, Hao ; MO, Kaichun ; GUIBAS, Leonidas J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, S. 652–660

[QYSG17] QI, Charles R. ; YI, Li ; SU, Hao ; GUIBAS, Leonidas J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: *Advances in neural information processing systems*, 2017, S. 5099–5108

[RA15] REDMON, Joseph ; ANGELOVA, Anelia: Real-time grasp detection using convolutional neural networks. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)* IEEE, 2015, S. 1316–1322

[RHW86] RUMELHART, David E. ; HINTON, Geoffrey E. ; WILLIAMS, Ronald J.: Learning representations by back-propagating errors. In: *nature* 323 (1986), Nr. 6088, S. 533–536

[RHW19] ROH, Yuji ; HEO, Geon ; WHANG, Steven E.: A survey on data collection for machine learning: a big data-ai integration perspective. In: *IEEE Transactions on Knowledge and Data Engineering* (2019)

[SEKB12] SAHBANI, Anis ; EL-KHOURY, Sahar ; BIDAUD, Philippe: An overview of 3D object grasp synthesis algorithms. In: *Robotics and Autonomous Systems* 60 (2012), Nr. 3, S. 326–336

[SLJ+15] SZEGEDY, Christian ; LIU, Wei ; JIA, Yangqing ; SERMANET, Pierre ; REED, Scott ; ANGUELOV, Dragomir ; ERHAN, Dumitru ; VANHOUCKE, Vincent ; RABINOVICH, Andrew: Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, S. 1–9

[STM+19] STAUB, Benno ; TANWANI, Ajay K. ; MAHLER, Jeffrey ; BREYER, Michel ; LASKEY, Michael ; TAKAOKA, Yutaka ; BAJRACHARYA, Max ; SIEGWART, Roland ; GOLDBERG, Ken: Dex-Net MM: Deep Grasping for Surface Decluttering with a Low-Precision Mobile Manipulator. In: *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)* IEEE, 2019, S. 1373–1379

[SZ14] SIMONYAN, Karen ; ZISSERMAN, Andrew: Very deep convolutional networks for large-scale image recognition. In: *arXiv preprint arXiv:1409.1556* (2014)

[TSK+18] TAN, Chuanqi ; SUN, Fuchun ; KONG, Tao ; ZHANG, Wenchang ; YANG, Chao ; LIU, Chunfang: A survey on deep transfer learning. In: *International conference on artificial neural networks* Springer, 2018, S. 270–279

[WARV12] WOHLKINGER, Walter ; ALDOMA BUCHACA, Aitor ; RUSU, Radu ; VINCZE, Markus: 3DNet: Large-Scale Object Class Recognition from CAD Models. In: *IEEE International Conference on Robotics and Automation (ICRA)*, 2012

[Wik20] WIKIPEDIA: *Normal distribution — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Normal%20distribution&oldid=944410789, 2020. – [Online; accessed 09-March-2020]

[WSK+15] WU, Zhirong ; SONG, Shuran ; KHOSLA, Aditya ; YU, Fisher ; ZHANG, Linguang ; TANG, Xiaoou ; XIAO, Jianxiong: 3d shapenets: A deep representation for volumetric shapes. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, S. 1912–1920

[YLWJ19] YOU, Kaichao ; LONG, Mingsheng ; WANG, Jianmin ; JORDAN, Michael I.: How Does Learning Rate Decay Help Modern Neural Networks? In: *arXiv e-prints* (2019), Aug, S. arXiv:1908.01878

[ZJK+19] ZHOU, Allan ; JANG, Eric ; KAPPLER, Daniel ; HERZOG, Alex ; KHANSARI, Mohi ; WOHLHART, Paul ; BAI, Yunfei ; KALAKRISHNAN, Mrinal ; LEVINE, Sergey ; FINN, Chelsea: Watch, Try, Learn: Meta-Learning from Demonstrations and Reward. In: *arXiv preprint arXiv:1906.03352* (2019)

[ZSY+18] ZENG, Andy ; SONG, Shuran ; YU, Kuan-Ting ; DONLON, Elliott ; HOGAN, Francois R. ; BAUZA, Maria ; MA, Daolin ; TAYLOR, Orion ; LIU, Melody ; ROMO, Eudald u. a.: Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In: *2018 IEEE international conference on robotics and automation (ICRA)* IEEE, 2018, S. 1–8

[ZZL+19] ZHANG, Hanbo ; ZHOU, Xinwen ; LAN, Xuguang ; LI, Jin ; TIAN, Zhiqiang ; ZHENG, Nanning: A Real-Time Robotic Grasping Approach With Oriented Anchor Box. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2019)