University of South Florida
Digital Commons @ University of South Florida

DIGITAL COMMONS
@ UNIVERSITY OF SOUTH FLORIDA

USF Tampa Graduate Theses and Dissertations | USF Graduate Theses and Dissertations

March 2023

A Human-in-the-Loop Robot Grasping System with Grasp Quality Refinement

Tian Tan
University of South Florida

Follow this and additional works at: https://digitalcommons.usf.edu/etd

 Part of the Robotics Commons

Scholar Commons Citation

Tan, Tian, "A Human-in-the-Loop Robot Grasping System with Grasp Quality Refinement" (2023). *USF Tampa Graduate Theses and Dissertations.*
https://digitalcommons.usf.edu/etd/9935

This Dissertation is brought to you for free and open access by the USF Graduate Theses and Dissertations at Digital Commons @ University of South Florida. It has been accepted for inclusion in USF Tampa Graduate Theses and Dissertations by an authorized administrator of Digital Commons @ University of South Florida. For more information, please contact digitalcommons@usf.edu.

A Human-in-the-Loop Robot Grasping System with Grasp Quality Refinement

by

Tian Tan

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Mechanical Engineering
College of Engineering
University of South Florida

Co-Major Professor: Rajiv Dubey, Ph.D.
Co-Major Professor: Redwan Alqasemi, Ph.D.
Sudeep Sarkar, Ph.D.
Kyle Reed, Ph.D.
Kandethody Ramachandran, Ph.D.

Date of Approval:
March 29, 2023

Keywords: Grasp Quality Measure, Grasp Pose Refinement
Computer Vision, Grasp Planning, Assistive Robots

Copyright © 2023, Tian Tan

Dedication

I dedicate my dissertation work to my family and friends.

Acknowledgments

I would like to express my deepest gratitude to my advisors, Dr. Rajiv Dubey and Dr. Redwan Alqasemi, for their support, guidance, and patience. I am also grateful to my dissertation committee, who generously provided advice and expertise. Additionally, I am thankful to the University of South Florida and the United States National Science Foundation for providing the resources and finance supporting my study and research.

Table of Contents

List of Tables

List of Figures

Abstract

The goal of this dissertation is to develop a grasping system for assistive robots that can help people with disabilities and the elderly to perform tasks of daily living. In developing this robot grasping system, we maximize its reliability, accuracy, and autonomy. High reliability and accuracy are required for robots to perform tasks around human users and to safely interact with objects that might be fragile or have contents that could spill. High autonomy is desired as users with disabilities are usually not dexterous enough to directly operate the robot. In this dissertation, a human-in-the-loop (HitL) robot grasping system is developed using computer vision and deep learning-based grasp detection. A grasp pose refinement module was created to finetune the grasp poses before robot execution to increase the grasping system's accuracy. A novel object movement-based grasp quality measure was developed to evaluate the grasp poses and provide indications for the grasp pose refinement. The grasping system includes the human user to supervise and provide corrections to increase reliability. While the default mode of the grasping system is highly automated, the user can directly control the robot in an assisted velocity control mode when the high autonomy mode fails. The scope of this work is object-oriented robot grasp planning, where we mainly use the object geometry features and the gripper-object contact properties to plan the object's grasp pose. The main contributions of this dissertation are the grasp quality measure and the grasp pose refinement method. The secondary contributions include the development of a vision module, an initial grasp detection module, an assisted velocity control module, and a graphical user interface module. We tested the grasp quality measure with a real robot grasping experiment and a method comparison experiment. The results of the robot grasping

experiment proved that the grasp quality scores calculated by the developed grasp quality measure closely match the real robot grasping results. The quality measure comparison experiment revealed that our grasp quality measure outperforms the state-of-the-art grasp quality measures in evaluating parallel-jaw gripper grasp poses when considering object movement. The HitL robot grasping system was tested by comparing its high autonomy mode with a state-of-the-art grasp detection network-based robot grasping system in real robot grasping. The results showed that our robot grasping system with the grasp refinement module works on a wider range of objects and could significantly improve the grasp success rate and reduce object movement compared to the grasp detection network-based robot grasping system.

Chapter 1: Introduction

1.1 Motivation

According to the World Health Organization (WHO), by 2050, the number of people aged 60 and older will reach 2.1 billion, which will be about 22% of the world's population. Compared to 12% in 2015, the elderly population will nearly double in 35 years. Besides population ageing, WHO also estimated that the number of people experience disability is over 1 billion, and this number is dramatically increasing [1]. Providing sufficient support, assistance, and services to improve the quality of life and well-being of the elderly and people with disabilities is a major challenge faced by all countries. Currently, people who need assistance in their daily lives rely on family, friends, and caregivers. There is a strong demand for technologies that can help them gain independence in their lives because many people who live their lives depending on assistants experience mental stresses in feeling being a burden on others. The World Report on Disability from WHO reported a survey of 1505 non-elderly adults in the United States with disability which showed that 70% of them relied on family and friends for daily living activities and 45% of them worried that caring for them would become too much of a burden on the family [2].

With the fast development of robotics, sensors, computer vision and machine learning technologies, assistive robots have become the most promising solution for helping people in need gain independence in their daily lives. And now, researchers have developed various assistive robotic systems [3, 4, 5] to assist people in different aspects of their lives. At the University of South Florida (USF) Center for Assistive, Rehabilitation and Robotics Technologies (CARRT), we are working on developing an assistive robot system to help individuals with diminished motor

skills perform activities of daily living (ADL)[1] independently. We selected a wheelchair-mounted robotic arm as the platform to implement and test our work. Our long-term goal is to develop a robot system that can gain autonomy through learning the environment and task information from the user's daily operations. Before the robot's learning capability is implemented, we need a starting system that can assist the user with simple daily tasks.

Because robot grasping is essential to most daily living tasks, in this dissertation, we aim to develop a human-robot collaborative system for grasping where human intelligence in decision-making is combined with the robot's perception, recognition, mobility, and dexterity. The developed robot grasping system has a high autonomy grasping mode and a low autonomy grasping mode. In the high autonomy grasping mode, the grasp planning and execution are performed by computer programs based on the input of an RGB-D camera. The human user is in the control loop to decide which object to grasp and supervise the autonomous programs to ensure they work correctly. In the low autonomy grasping mode, the human user directly controls the velocity of the robot's end-effector using human-eye visual feedback with an assisted velocity control method. Note that the grasping system developed in this work can be used not only for assistive robots but also for other robot applications where a human operator is present and accurate grasping is required.

1.2 Problem Statement

The goal of this dissertation is to develop a human-in-the-loop robot grasping system with low- and high-autonomy grasping modes. The high autonomy grasping mode is based on a hybrid grasp planning method which contains a grasp detection network for initial grasp pose detection, and a grasp pose refinement module for fine-tuning all six degrees of freedom of the initial grasp

[1] This work is supported by NSF (award number: 1826258).

pose. We developed a novel grasp pose refinement method with a novel grasp quality measure to achieve collision-free and object movement-minimized grasping. The grasp quality measure is used in refining the grasp pose's projection parameters (x, y, θ_z). This grasping mode has high autonomy, a high grasp success rate, minimized object movement, and minimized chance of collision or empty grasp. The main limitations are that the system relies heavily on the accuracy of the depth data from the camera and is designed for only parallel-jaw gripper grasping. This work is within the category of object-oriented robot grasp planning, where we mainly use the object geometry features and the gripper-object contact properties to plan the object's grasp pose. After obtaining the grasp pose, the robot motion for performing the grasp is planned using MoveIt [6], the robot motion planning framework.

The low autonomy grasping mode uses camera depth data, virtual fixture, mixed coordinate systems, and automatic mode switching to allow the user to use three degrees of input to control the six-dimensional grasp pose plus one more dimension for opening or closing the gripper. Compared to the traditional direct mapping and mode-switching control methods, the novelty of this work is the derivation of the kinematics of a virtual fixture-based velocity control scheme and the design of the grasp pose parameters' controlling order and the automatic mode switching. This grasping mode is vision-based teleoperation that needs more user input but is more reliable and accurate. The only limitation of this grasping mode is the accuracy of the depth data, which directly affects the establishment of the virtual fixture used in assisting the user's control.

Five main modules were developed to construct the described robot grasping system: the interface module, the robot vision module, the grasp detection module, the grasp refinement module, and the assisted velocity control module. The interface module contains a graphical user interface to help the user supervise and operate the robot and a main script that manages the

system's workflow. The robot vision module processes the RGB-D camera data and provides useful visual information for the other modules. The grasp detection module is based on an existing grasp detection neural network which generates the initial grasp pose. The grasp refinement module uses point cloud data and our object movement-based grasp quality measures to fine-tune the grasp pose. The assisted velocity control module is for the low autonomy grasping mode.

1.3 Objectives

After establishing the design of the desired robot grasping system, the corresponding objectives of this work are the following:

1. Develop an object movement-based grasp quality measure to predict the object movement in grasping and evaluate the quality of grasps.
2. Develop a robot vision module based on a state-of-the-art object recognition and segmentation neural network and techniques for point cloud processing to obtain visual information of the object and scene.
3. Develop a grasp detection module based on a state-of-the-art grasp detection neural network to generate the initial grasp poses ($x_i, y_i, z_i, \theta_{xi}, \theta_{yi}, \theta_{zi}$).
4. Create a grasp refinement module to improve the grasp quality of the initial grasp pose generated by the grasp detection module. This work includes: (a) deriving the grasp direction (pitch θ_x, yaw θ_y) and depth (z) refinement criteria using the geometry information computed from the object and scene point cloud; (b) designing the refinement procedure for the grasp pose projection parameters (x, y, θ_z) based on the object movement-based quality measure and the object surface geometry information captured by the vision module.
5. Develop a graphical interface to provide feedback to the user and a main script that manages the system's workflow.

6. Conduct experiments to demonstrate the effectiveness of the object movement-based grasp quality measure and high-autonomy grasping mode of the human-in-the-loop robot grasping system.

In addition to the list of objectives, the main components of the developed robot grasping system and the outline of this dissertation are shown in Figure 1.1.

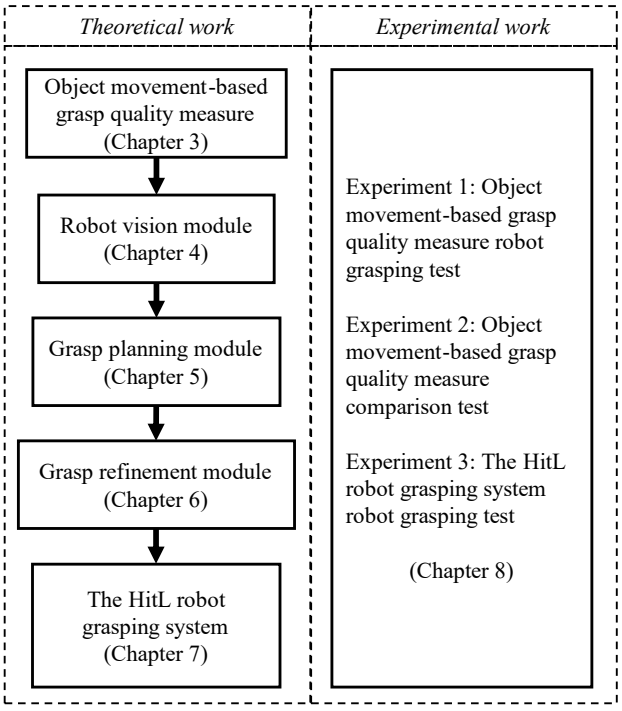


Figure 1.1 Block diagram of the main components in the developed robot grasping system.

1.4 Contributions

In this work, there are two main contributions (the first two in the following list) and four secondary contributions, which are listed as the following:

1. A novel grasp quality measure that outperforms existing methods for evaluating parallel-jaw gripper grasp poses (or antipodal grasp poses) when considering object movements in grasping.

The developed grasp quality measure was published in 2021 RA-L and presented in 2021 IROS titled “Formulation and Validation of an Intuitive Quality Measure for Antipodal Grasp Pose Evaluation” [7].

2. A novel grasp refinement method that can improve the robot grasping performance in avoiding collision or empty grasp, forming better gripper-object contact, and minimizing the object’s movement. A manuscript titled “A Novel Grasp Pose Refinement Method for Robot Grasping that Minimizes Object Movement” was submitted to 2023 IROS.
3. A robot vision module developed from customizing and integrating different existing works to provide the visual information required by our robot grasping system.
4. A grasp pose filtering method for the grasp detection module. This method works better than the default score-based pose selection method in selecting grasp poses from the grasp detection neural network output.
5. An assisted velocity control method for robot grasping. This method’s development includes deriving the virtual fixture-based velocity control kinematics and designing the automatic switching of control modes. A manuscript about this velocity control method is in preparation.
6. A computer vision-based human-in-the-loop robot grasping system with high and low autonomy grasping modes. The robot grasping system’s high autonomy grasping mode can work on a broader range of objects and achieve a higher grasp success rate and less undesired object movement than a state-of-the-art grasp detection network-based grasping system. The low autonomy grasping mode uses an intuitive velocity control method.

Chapter 2: Background and Related Work

Finding an appropriate end-effector pose for a robot to grasp an object (often called robot grasp synthesis/detection/planning) is a fundamental but challenging problem in robotics. Numerous research has been conducted to tackle the robot grasp planning problem over the past several decades. In the early years, the field of robot grasping was dominated by analytical approaches, which synthesized robot grasp poses based on the grasping mechanics and the gripper-object interaction properties [8, 9]. The old analytical approaches had minimal usage in real-life robot grasping because they rely on prior knowledge of the object (such as the object's geometric and physical properties), they usually oversimplify the gripper-object interactions, and their theoretical models of robot grasping were hard to generalize. To overcome the shortcomings of the analytical approaches and develop better techniques to tackle the problem of unknown object grasping, researchers started to explore the data-driven (empirical) approaches around the year 2000. The core idea of the data-driven approach is to use the obtained grasping data (experience) to infer the grasp poses for known, similar, and unknown objects. The traditional data-driven approaches achieve grasp synthesis based on classical machine learning techniques. There are different types of traditional data-driven approaches, and the readers are referred to references [9, 10, 11] for a comprehensive review. With the rapid development of deep learning, deep convolutional neural network-based grasp detection (the modern data-driven approach) started to show promising results in robot grasp planning. In the year 2013, Lenz et al. [12] published the pioneering work of using deep learning to tackle the grasp planning problem. And now, the field of robot grasp planning is dominated by deep learning-based approaches [13, 14, 15].

The state-of-the-art grasp detection deep neural networks have excellent generalizability and efficiency, which can generate high success rate grasp poses in milliseconds. However, the weakness of the grasp detection networks is that the detected grasp poses are often not optimized, which is a shared weakness of all data-driven approaches. When the grasp poses are not optimized, undesired object movements can occur during grasping. The undesired object movements will change the object pose, affecting the post-grasp task, spilling the content in a grasped container, and forming unstable grasps. Therefore, considering object movement in grasp planning is essential, especially for assistive robots. In this work, we developed a hybrid robot grasp planning approach for our robot grasping system's high-autonomy mode. This hybrid approach uses a grasp detection network for initial grasp pose generation, and an analytical grasp pose refinement module to improve grasp accuracy (minimizing object movement and avoiding collision or empty grasp). In addition, we developed an assisted velocity control-based low autonomy grasping method to allow the user to teleoperate the robot for grasping with minimum required operation input.

2.1 Deep Learning-Based Grasp Detection

Based on the grasp pose representations, the grasp detection networks can be classified into two categories: the grasp rectangle-based and the 6D grasp pose-based. The grasp rectangle-based approaches [16, 17, 18, 19, 20, 21, 22] take RGB or RGB-D images as input and detect robot grasp poses in the image space as grasp rectangles. A grasp rectangle is fully defined by five parameters, two for its planar position, one for its planar orientation, and two for its size. The grasp rectangle-based approaches are limited to detecting grasp from the direction perpendicular to the image plane. The camera must be set with a fixed distance to the object for those treating one [19, 20] or both [18, 21] size parameters of the grasp rectangle as constants. Therefore, rectangle-based grasp detection requires an additional process for view-direction determination [22] in general grasping

scenarios. In contrast to the grasp rectangle-based approaches, the 6D pose-based approaches are not limited to approach direction-fixed grasping. In recent years, 6D pose-based approaches [23, 24, 25, 26, 27] were developed and became the most popular in robot grasp detection. The 6D grasp pose detection networks take the RGB-D image or the point cloud as input and output the 6D grasp poses [23, 27] or 7D grasp configurations (6D pose plus gripper open width) [24, 25, 26]. Adding the gripper open width to the grasp detection increases the robustness of the network in detecting collision-free poses. Our work used the Contact-GraspNet (CGN) [23] as the grasp detection module's backbone for initial 7D grasp configuration generation.

Besides the grasp representation-based method classification, it is also important to point out that the modern grasp detection approaches are mainly used for unknown object grasping; and the known object grasping problem is diverted to the object 6D pose estimation problem. Since optimal grasp poses can be pre-determined and stored with the object's model/template, finding the grasp pose is equivalent to finding the object's 6D pose relative to the robot [28, 29]. The object's 6D pose estimation can be classified into instance-level (using exact object models) [28] and category-level (using one model template for all objects of the same category) [29]. In this work, we used the point pair feature-based surface matching [30, 31] to estimate the poses of known objects (instance-level) for grasping. Compared to the category-level approach, the instance-level approach requires more effort to prepare the object dataset, but it can yield more accurate object models and corresponding grasp poses. The readers are referred to [32, 33] for thorough reviews of the object 6D pose estimation methods.

2.2 Grasp Pose Refinement

As the robot grasp poses detected by grasp detection networks are not optimized, researchers started to explore different methods to improve the grasp pose accuracy and achieve

collision-free grasp. Ainetter et al. [16] presented a Multilayer Perceptron (MLP)-based grasp pose refinement for the grasp rectangle-based grasp detection network. Wei et al. [26], Ni et al. [34], and Cai et al. [35] implemented different grasp pose refinement methods to improve the performance of 6D grasp detection networks. Wen et al. [29] used grasp pose refinement to fix the pre-defined grasp poses' distortion caused by the deformation of the object model template. In category-level object pose estimation, the object model template usually needs to be deformed to match the actual object point cloud. Our hybrid grasp planning approach is similar to [26, 34, 35] in terms of the fundamental structure, which is a grasp detection neural network (for high-efficiency grasp pose generation) combined with a grasp refinement module (for local optimization of the grasp accuracy). Wei et al. [26], Ni et al. [34], Cai et al. [35], and our approach used the PointNet++ [36], the SPH3D-GCN [37], a customized light-weight volume-point network (VPN) [35], and the Contact-GraspNet [25] as the backbone for generating initial grasp proposals, respectively. Because these 6D/7D grasp detection networks use point cloud data as input, they cannot detect grasp for objects with similar depth to the background. For example, these networks cannot detect any grasp pose for a thin cell phone lying on a table. While this special case is not considered in [26, 29, 34, 35], our work uses Detectron2 [38] to detect small or thin objects, and grasp pose can be planned using the detected object silhouette and our grasp refinement module.

The grasp refinement modules of [16, 26, 34] are neural network-based and trained with the grasp detection network in an end-to-end fashion; the grasp refinement modules of [29, 35] are also neural network-based but trained as an independent network from the pose detection network; our grasp refinement is analytical. The end-to-end network approach maximizes the efficiency of the grasp detection and refinement process; however, it is difficult to obtain optimized grasp poses using this data-driven approach. Because the initial grasp detection and our analytical grasp pose

refinement are fast enough for real-life robot applications, we focus on maximizing the generated grasp pose accuracy (or minimizing the object movements during gripper closing) in this work.

Our analytical grasp refinement is comprehensive and intuitive because every parameter of the 7D grasp configuration is finetuned, and the refinement guidelines are all human-understandable gripper-object interaction properties. Based on the mechanics of parallel-jaw gripper grasping, the seven grasp configuration parameters, when referenced to the gripper frame (${}^{G}x$, ${}^{G}y$, ${}^{G}z$, ${}^{G}\theta_x$, ${}^{G}\theta_y$, ${}^{G}\theta_z, w$), can be interpreted as four independent parts: the grasp direction (approach vector) (${}^{G}\theta_x$, ${}^{G}\theta_y$), the grasp depth (${}^{G}z$), the grasp pose projection in the direction of the grasp approach vector (${}^{G}x$, ${}^{G}y$, ${}^{G}\theta_z$), and the gripper open width (w). In grasp direction refinement, the gripper pitch (${}^{G}\theta_x$) is optimized to make the approach vector as close to perpendicular as possible to the object surface; the gripper yaw (${}^{G}\theta_y$) is refined to ensure stable contact between the gripper and the object. The grasp depth is optimized to ensure enough object surface is grasped (stable grasp) and avoid collisions or empty grasps. The gripper's open width is optimized to prevent the gripper from colliding with other objects near the target object. The grasp pose's projection parameters are refined based on our object movement-based grasp quality measure [7].

2.3 Grasp Quality Measure

The traditional grasp quality measures are developed based on the force-closure and form-closure properties of robot grasping [39]. The Q-measure [40] and many variants of it are the most used analytical grasp quality measures in existing works, for example, the GraspIt! simulator [41] uses the Q-measure for grasp synthesis. In recent works, deep learning-based grasp detection often generates quality scores along with the output grasp poses. Reviews of the grasp quality measures can be found in [9, 42]. Our object movement-based grasp quality measure is developed based on

the object movement features extracted from analyzing the gripper-object interactions. While the mentioned grasp quality measures have different pros and cons, our object movement-based quality measure best fits our grasp refinement module since we prefer grasps that cause minimum object movements. A detailed comparison between the representative existing grasp quality measures and our quality measure is presented in Chapter 8.

In robot grasping experiments, the main metric used for measuring the performance (grasp quality) is the grasp success rate which is an effective and straightforward indicator. However, when studying the actual grasp quality, the success rate measurement is insufficient because it cannot differentiate the quality of grasps that are all recorded as successful. The success or failure of a grasp is often measured as whether the grasp can pick up the object or not, and for the grasps that can pick up the object, their actual quality can also be different. In our robot grasping experiment, we created a novel method to measure the object movement caused by the gripper during grasping, which is an effective continuous measure of the grasp pose accuracy or quality. In addition, we also introduced the observed grasp result measure to capture the successful but unstable grasps. When the robot picks up the test object with an unstable grasp, its measured grasp result is a success, but its observed grasp result is a failure. With the new measurement in our robot experiment, our results can show the grasp accuracy improvement after pose refinement more clearly than other related works [16, 26, 29, 34, 35].

Chapter 3: Object Movement-Based Grasp Quality Measure[2]

This chapter presents a novel object movement-based grasp quality measure developed to evaluate the parallel-jaw gripper grasp pose and provide the rationale for the grasp projection parameters' refinement process, an important process in the grasp refinement module (Chapter 6). Until now, most researchers have evaluated robot grasping based on only the robot's success rate of picking up objects and overlooked the objects' movement during gripper closing. While in real-world assistive robot applications, one must consider object movement in robot grasping. If an object is moved during grasping, it could affect the object's surroundings, the content inside the object, and the object's pose information will need to be updated for post-grasp tasks. For example, Figure 3.1 (a) shows a robot gripper grasping a toothbrush near a cup. The robot successfully grasped the toothbrush, but when the gripper fingers closed on the toothbrush, the toothbrush rotated and knocked over the cup. Similarly, Figure 3.1 (b) shows a toy banana grasping example in which the toy banana rotated and slid during gripper closing. The change of the banana's in-hand pose can cause the failure of the post-grasping task, such as peeling the banana. Therefore, the grasp success rate is a necessary but insufficient condition for a good grasp in assistive robotics; and the object movement must be considered in grasp evaluation and planning.

When a parallel-jaw gripper grasps an object, it can cause the object to translate in the gripper closing direction, rotate about the gripper z-axis, and slide in the direction perpendicular to the gripper closing direction. The grasp quality is negatively correlated to the amount of object

[2]This chapter was published in IEEE Robotics and Automation Letters, vol. 6, no. 4, pp. 6907-6914, Oct. 2021, doi: 10.1109/LRA.2021.3096192. Permission is included in Appendix A.

movement during grasping. Based on these three types of object movement, four grasp quality-related features were designed: the translation predictor, the rotation predictor, the type-I slippage predictor, and the type-II slippage predictor. In addition, another two quality features, the missing and collision indicators, were added for the special cases where the gripper misses or collides with the object, respectively.

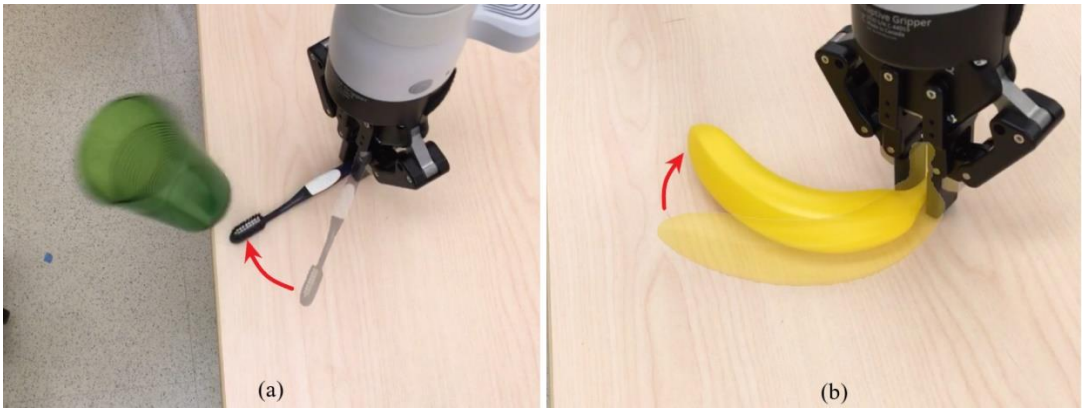


Figure 3.1 Examples of robot grasps that cause significant object movement during gripper closing.

The grasp quality score is calculated as the weighted sum of the normalized scores of the quality features. The feature score normalization and weight assignment were tuned through experiments so that the final grasp quality score can approximate the actual robot grasping success rate. In this way, the grasp quality score indicates not only the likelihood of the robot picking up the object, but also measures how much the object might move during grasping.

The schematic diagram of this quality measure is shown in Figure 3.2. The quality measure takes the 2D representations of the robot gripper and the object as input and outputs the normalized quality score of the input grasp pose. The grasp quality score is set to zero if the grasp misses or collides with the object. Otherwise, the grasp quality score is calculated using the object

movement-based grasp quality features extracted from the two low-level visual features, the grasp region profile vector (GRPV) and the contact region feature vector (CRFV).

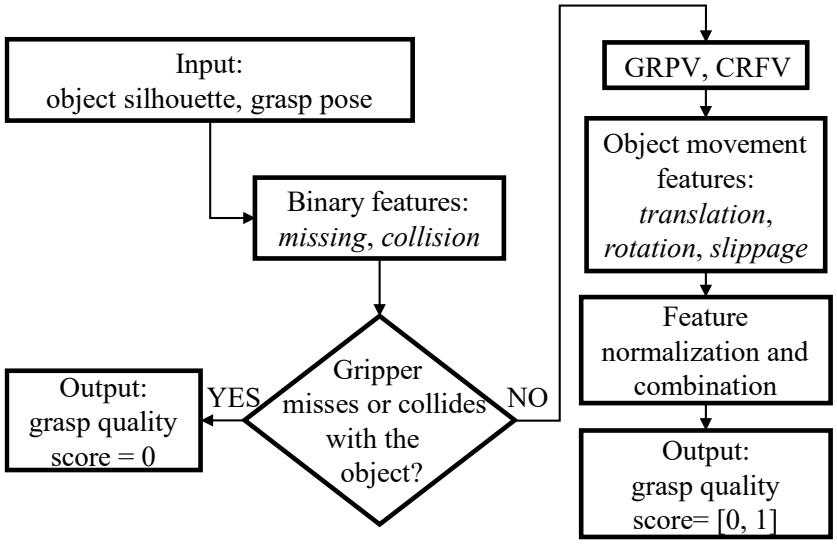


Figure 3.2 Grasp quality measure schematic.

3.1 Gripper and Object Representations

The gripper and object must be represented mathematically to extract the grasp quality features. Various gripper and object representations have been used in developing quality evaluation algorithms in this field. The robot gripper is often represented by the contact surfaces of its fingers. A contact surface can be approximated as a point [43], a line [44], a circle [45], or a gridded surface [46]. As for the object, the most common representations are object contour (or silhouette) [43], object partial point-cloud [47], or object 3D model [48]. In this dissertation, we address the parallel-jaw gripper grasping problem with the object described as its silhouette and the gripper described using a novel grasp line representation.

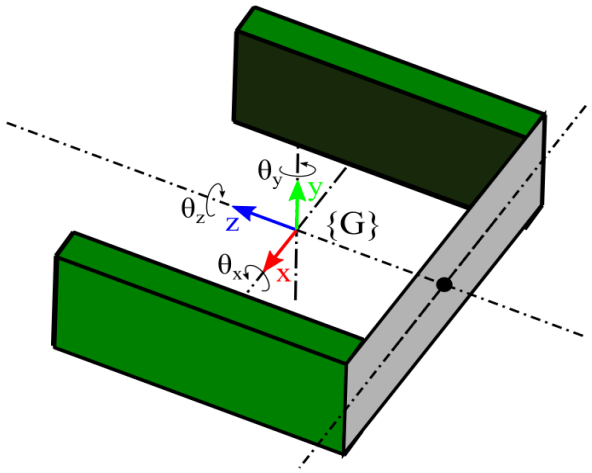


Figure 3.3 Illustration of the gripper frame assignment.

As shown in Figure 3.3, the pose of a parallel-jaw gripper is fully defined by 6 Cartesian coordinates (x, y, z, pitch: θ_x, yaw: θ_y, roll: θ_z). These coordinates can be classified into grasp pose projection parameters (x, y, roll), grasp direction parameters (pitch, yaw), and grasp depth (z). The grasp direction parameters define the direction of the gripper's z-axis. Usually, for the same grasp direction, there are grasp poses of different qualities (e.g., two grasp poses of the same grasp direction, one is centered to the object while the other is not). For different grasp directions, grasp poses of the same quality can also exist (e.g., grasp a sphere from different directions). Therefore, the grasp direction parameters cannot be directly used to infer the grasp quality. The grasp depth describes how deep the gripper grasps the object in the grasp direction. If the grasp depth is too shallow, then the gripper will miss the object; if the grasp depth is too deep, then the gripper could collide with the object or other objects/surfaces around the object; if the grasp depth is in a proper range, then it does not affect grasp quality. The grasp direction parameters and the grasp depth are related to the grasp quality in a discontinuous manner, and they should be evaluated based on the task requirements and the grasp depth requirement, respectively. This chapter focuses

on developing a continuous quality measure based on the grasp pose projection parameters (x, y, roll). Because of the mechanics of the parallel-jaw gripper grasping, we can simplify the gripper and object representations by projecting both the gripper and the object orthogonally to an image plane in the grasp direction (yaw, pitch) with the grasp depth (z) as the projection distance. In this way, the gripper and object can be represented by the grasp rectangle and the object silhouette in the projection image. As shown in Figure 3.4, the two highlighted opposing edges of the grasp rectangle correspond to the two gripper fingers' contact surfaces.

3.1.1 Mathematical Description of the Gripper Projection

We designed a more detailed mathematical description for the gripper projection than just a grasp rectangle. The gripper projection is modeled as a set of line segments, $\{gl_1, gl_2, \ldots, gl_n\}$, representing the discretized gripper closing path. As shown in Figure 3.4, the dashed lines are the grasp lines. In the projected-gripper frame {G}, let point P_{ij} be the j^{th} point on grasp line i (gl_i), then the function of gl_i can be expressed as equations (3.1) and (3.2).

$$P_{ij}^{x} = x - \Delta w_i \sin\theta + \Delta r_j \cos\theta \tag{3.1}$$

$$P_{ij}^{y} = y - \Delta w_i \sin\theta + \Delta r_j \cos\theta \tag{3.2}$$

$$where\ \Delta w_i = \frac{w(2i - n - 1)}{2(n - 1)} \tag{3.3}$$

where i (1, 2, …, n) is the index of a grasp line, Δw_i is the distance from the center of the gripper projection to the i^{th} grasp line, Δr_j is the distance from a grasp line's center to its point j. Also, (Δr_j , Δw_i) is the coordinate of P_{ij} in the projected-gripper frame {G}. Δr_j is positive if P_{ij} is on the right side of the grasp line center, and negative if otherwise ($\Delta r_j \in \left[-\frac{r}{2}, \frac{r}{2}\right]$). The parameters

of this grasp representation, (x, y, θ, r, w, n), are composed of the planar center location (x, y), planar orientation (θ), gripper finger width (w), gripper open width (r), and the number of grasp lines $(n \in [1, w])$ of the gripper projection in the image coordinates.

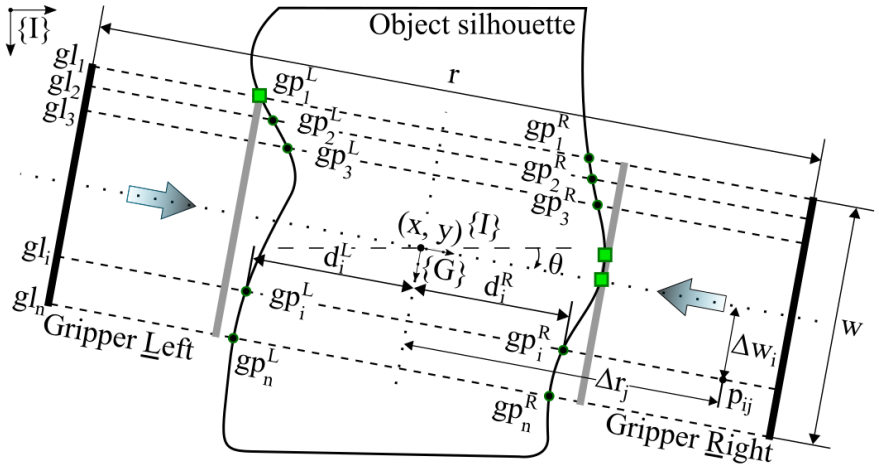


Figure 3.4 Illustration of grasp lines (gl_i), grasp points (gp_i^e), grasp distances (d_i^e), contact points and grasp pose parameters (x, y, θ, r, w, n). The thicker lines labeled as *L* and *R* represent the left and right gripper fingers' contact lines. The arrows indicate the gripper closing direction.

In figures and equations, superscripts *L* and *R* are used to differentiate the gripper's left and right sides. A term with superscript *e* indicates that the term is side specific, and the *e* should be replaced by either *L* or *R*. The intersections of the grasp lines and the object silhouette outline are referred to as the *grasp points* (gp_i^e) of the object. The distances between grasp points and the center points of the corresponding grasp line segments are referred to as the *grasp distances* (d_i^e) of those grasp points. When the gripper closes along the grasp line, some of the grasp points will contact the gripper, and these grasp points become *contact points*. The green square shape grasp points in Figure. 3.4 are the contact points corresponding to that grasp pose. The collection of grasp points is the *grasp region*, and the collection of contact points is the *contact region*.

3.1.2 Mathematical Description of the Object Projection

Object projection is the specialized object silhouette for grasping. It is a binary image/matrix where the pixel values/matrix entries are 1's for the object, and 0's for the background. Unlike the regular object silhouette, which represents the outline of the whole object, the silhouette for grasping must be able to represent the object's contact surface in grasping. As shown in Figure 3.5, there are two silhouettes of a mug viewed from the top (left image). The silhouette in the middle is the regular silhouette, which is a solid shape of the mug's outline. The silhouette on the right is the silhouette for grasping, which captures not only the outline shape of the mug but also the inside geometry. When the robot grasps the body of the mug by the outermost surface, both silhouettes in Figure 3.5 can represent the geometry of the contact surface. However, when the robot grasps the mug by a single side of the mug wall, only the silhouette for grasping on the right side can represent the mug's contact surface. In Figure 3.6, there are two silhouettes of a mug viewed from the side (left image). The silhouette in the middle represents the body of the mug, and the one on the right represents the handle of the mug. If the robot grasps the mug by the body of the mug, then the silhouette for grasping is the middle one; and if the robot grasps the mug by the handle, then the silhouette for grasping is the right one.

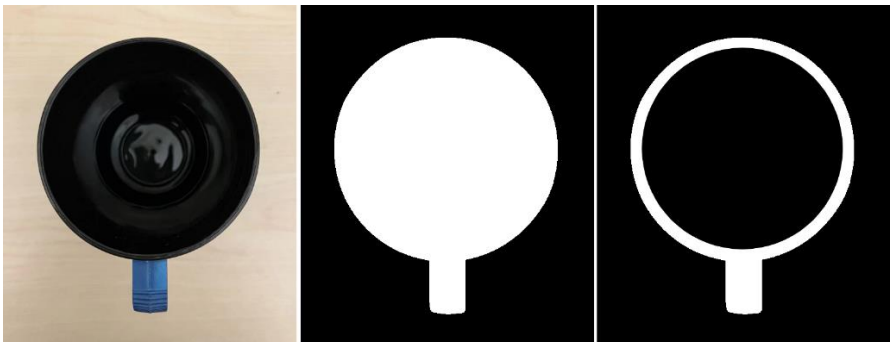


Figure 3.5 Two types of silhouettes for a mug viewed from the top. The middle is the regular silhouette; the right is the silhouette for grasping.

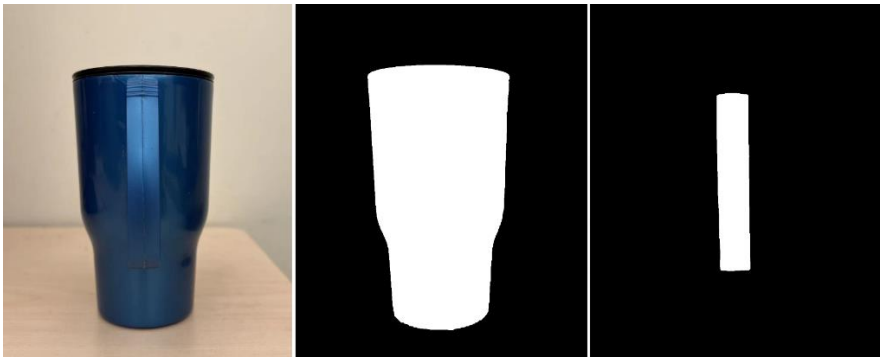


Figure 3.6 Two types of silhouettes for a mug viewed from the side. The middle is the silhouette for grasping when the mug's body is grasped; the right is the silhouette for grasping when the mug's handle is grasped.

In summary, the object's silhouette for grasping depends on both the object geometry and the grasp pose; It should clearly show the object's contact surface corresponding to the given grasp pose. This dissertation used two methods to extract the object silhouette for grasping. Suppose the object silhouette for grasping is the same as the regular silhouette. In that case, the object mask generated by the object recognition and segmentation neural network (Chapter 4) can be used as the object's silhouette for grasping. For other cases, an object point cloud-based silhouette extraction method was used (Chapter 6).

3.2 Grasp Quality Features

Based on the mathematical representations of the object and grasp pose, a set of grasp features, which include two low-level visual features and six high-level quality features, were developed. The visual features contain rich information about how the gripper will interact with the object, and the quality features are intuitive quality measures of the grasp.

3.2.1 Low-Level Visual Features

The visual features are the grasp region profile vector (GRPV) and the contact region feature vector (CRFV), which carry information about the geometry and the contact properties of

the object contour inside the grasp region. The GRPV is a vector of the grasp distances of all grasp lines. It can be expressed in mathematical terms as the following equation:

$$GRPV = \begin{bmatrix} GRPV^L \\ GRPV^R \end{bmatrix} = \begin{bmatrix} d_1^L & d_2^L & \cdots & d_n^L \\ d_1^R & d_2^R & \cdots & d_n^R \end{bmatrix} \tag{3.4}$$

The grasp distances (d_i^L or d_i^R) are found by searching the left and right outermost intersections of the object outline and the grasp lines. With the parameters of the input grasp pose (x, y, θ, w, r, n) the grasp lines are defined. The intersection searching points are calculated using the grasp line equations (3.1) and (3.2) with Δ_{r_j}as the only variable. When the searching points satisfy the left and right outermost intersections' property conditions, the intersections are found, and their corresponding searching variables are the grasp distances of the grasp line. The searching points' pixel values are monitored in finding the intersections. An intersection point must satisfy two conditions: (a) its pixel value is 1, (b) at least one of its two neighboring points on the same grasp line has a pixel value of 0. I.e., the pixel values of an intersection (underlined) and its two neighboring points must be [0, 1, 1] (left contact), [1, 1, 0] (right contact), or [0, 1, 0] (left and right contacts). To ensure the intersections found are the outermost intersections, we start searching from the outermost point of each side.

For each grasp line, if any one of the endpoints has a pixel value of 1, then the gripper will collide with the object, the *collision indicator (I_c)* will be set to 1, and the extraction of other features can stop. If the grasp passes the collision checking, a coarse search that uses d_{step} as the step size is carried out to find a point of pixel value 1 on each side of the grasp line. Then a fine backward search with step size 1 is used to find the contact point. The maximum number of searches can be calculated using equation (3.5).

$$N_{max}^{s} = \frac{r}{d_{step}} + d_{step} - 1 \tag{3.5}$$

where $\frac{r}{d_{step}}$ and $d_{step} - 1$, are the maximum number of searches in the coarse and fine searching processes, respectively. The derivative of the total maximum searches with respect to the step size d_{step} is:

$$\frac{dN_{max}^{s}}{dd_{step}} = 1 - \frac{r}{d_{step}^{2}} \tag{3.6}$$

Let equation (3.6) equal to zero, which yields $d_{step} = \sqrt{r}$, the optimal step size for minimizing the maximum number of searches. If no grasp point is found after searching all grasp lines, the GRPV will be an empty vector, and the *missing indicator (I_m)* is set to 1. I_c and I_m are 0's by default, and this is the only case that the high-level movement features need to be calculated. In other cases, the robot gripper will either miss ($I_m = 1$) or collide ($I_c = 1$) with the object. Figure 3.7 shows how the GRPV feature captures the object's grasp region shape, and the GRPV is visualized by drawing lines that have the length of the grasp distances along the grasp lines.

After calculating the GRPV, the left and right *primary contact points* can be defined as the grasp points with maximum absolute grasp distances in the left and right grasp regions, respectively. If the grasp distances' difference between a grasp point and the primary contact point of the same side is below a threshold, this grasp point is a *secondary contact point* on that side. Adding secondary contact points into consideration decreases the effect of false contact prediction caused by imperfect object silhouette detection. Once the contact points are found, we can calculate their normals and center offsets and organize them in the CRFV as in equation (3.7).

$$CRFV = \begin{bmatrix} CRFV^L \\ CRFV^R \end{bmatrix} = \begin{bmatrix} CPN^L \\ CPD^L \\ CPN^R \\ CPD^R \end{bmatrix} \tag{3.7}$$

where CPN^e and CPD^e are the vectors of contact point normal and contact point center offsets, respectively. Examples of the contact point normals can be seen in Figure 3.7. The contact point center offset equals the distance from the gripper center to the grasp line that the contact point is on, which is the Δw_i in equation (3.3).

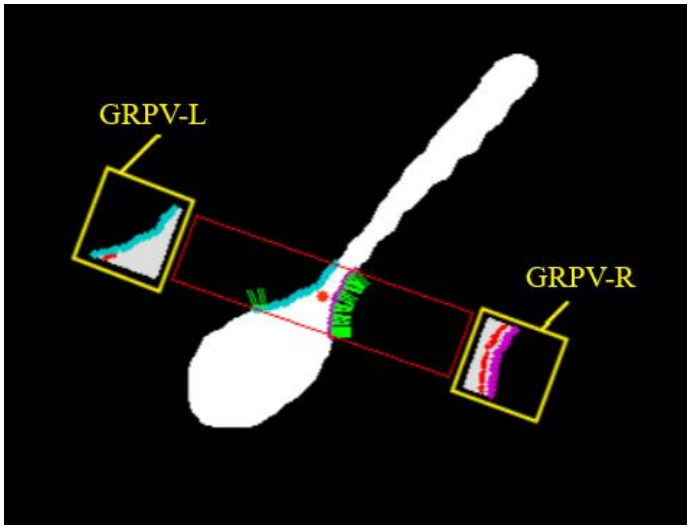


Figure 3.7 Visualization of low-level features extracted from the spoon silhouette and a given grasp pose(red rectangle). The GRPV is drawn as a collection of line segments on each side of the grasp rectangle. The left grasp region(turquoise), the right grasp region(purple), the contact region(red points in GRPV), and the contact region normals (green lines) are highlighted in different colors.

We developed a surface normal approximation method to extract the normals of contact points. As shown in Figure. 3.8, the normal of an object boundary pixel can be approximated by inverting the sum of its non-vacant neighboring vectors. Here, a neighboring vector (V_{ni}) is a

vector from the pixel of interest to one of its neighboring pixels, and a neighboring pixel is non-vacant if its pixel value equals 1. Moreover, the shape of the neighborhood is an essential factor affecting the approximation result, and it should be a circle since the neighbors in a circle are evenly distributed in all directions. Therefore, we assign weights to the square window for the neighborhood to form a circular shape. The calculation procedure of this normal extraction method is presented in equations (3.8) to (3.10).

$$N = normalize\left[-\left(\sum_{i}^{m}\sum_{j}^{m} M_{xij}\,,\sum_{i}^{m}\sum_{j}^{m} M_{yij}\right)\right] \tag{3.8}$$

$$M_x = M_{x0} \circ M_p \circ M_w \tag{3.9}$$

$$M_y = M_{y0} \circ M_p \circ M_w \tag{3.10}$$

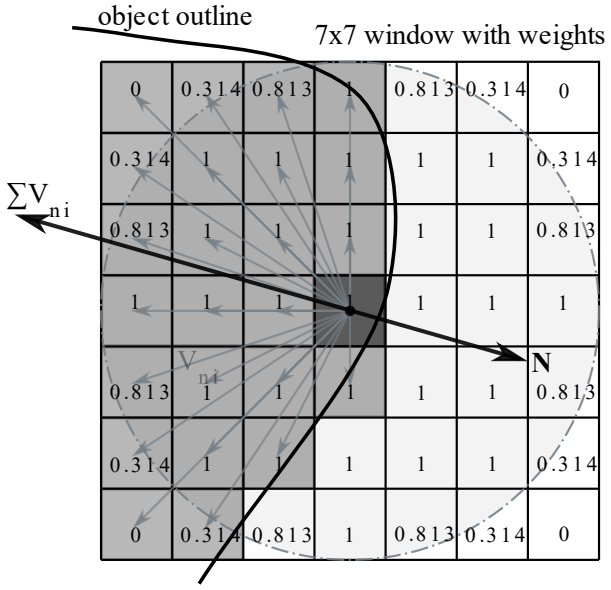


Figure 3.8 The visualization of object silhouette boundary pixel normal approximation. N is the pixel normal, V_{ni}'s are the neighboring vectors.

In equation (3.8), N is the normal of the object boundary point, and m is the size of a square window. M_x and M_y are the matrices of the weighted x and y coordinates of the non-zero pixels inside the window. The sum of all entries in M_x and M_y, respectively, are the x and y coordinates of the sum of all non-vacant neighboring vectors, and the normal *N* is in the opposite direction of this vector. In equation (3.9) and (3.10), M_{x0} and M_{y0} are the matrices of the original x and y coordinates of the corresponding pixels in the $m \times m$ window that centered at the point of interest. M_p is the matrix of pixel values of the pixels inside the window, which is a binary masking matrix. M_w is the weight matrix of the window. These calculations are performed in the window coordinate system, which has the point of interest as the origin and the same orientation as the image coordinate system. Equations (3.11) and (3.12) show the mathematical expressions of M_{x0} and M_{y0}.

$$M_{x0} = \begin{bmatrix} -3 & -2 & -1 & 0 & 1 & 2 & 3 \\ -3 & -2 & -1 & 0 & 1 & 2 & 3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -3 & -2 & -1 & 0 & 1 & 2 & 3 \end{bmatrix}_{7\times7} \tag{3.11}$$

$$M_{y0} = \begin{bmatrix} -3 & -3 & \cdots & -3 \\ -2 & -2 & \cdots & -2 \\ -1 & -1 & \cdots & -1 \\ 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 \\ 2 & 2 & \cdots & 2 \\ 3 & 3 & \cdots & 3 \end{bmatrix}_{7\times7} \tag{3.12}$$

To test the algorithm, 3, 5, 7, and 9 were used as the window size *m*, the best normal approximation results are obtained using *m* equals 7, and the corresponding weight matrix M_w is shown in equation (3.13):

$$M_w = \begin{bmatrix} 0 & 0.314 & 0.813 & 1 & 0.813 & 0.314 & 0 \\ 0.314 & 1 & 1 & 1 & 1 & 1 & 0.314 \\ 0.813 & 1 & 1 & 1 & 1 & 1 & 0.813 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.813 & 1 & 1 & 1 & 1 & 1 & 0.813 \\ 0.314 & 1 & 1 & 1 & 1 & 1 & 0.314 \\ 0 & 0.314 & 0.813 & 1 & 0.813 & 0.314 & 0 \end{bmatrix} \quad (3.13)$$

Figure 3.9 shows the normal extraction results of a complex random geometry using the presented normal extraction method. The green lines are the estimated normals, and it is easy to see that the estimation works well. It is also worth mentioning that even though we only used this method to extract boundary normal in 2D image space, this method can also be used to estimate the surface normal of 3D objects represented by voxels.

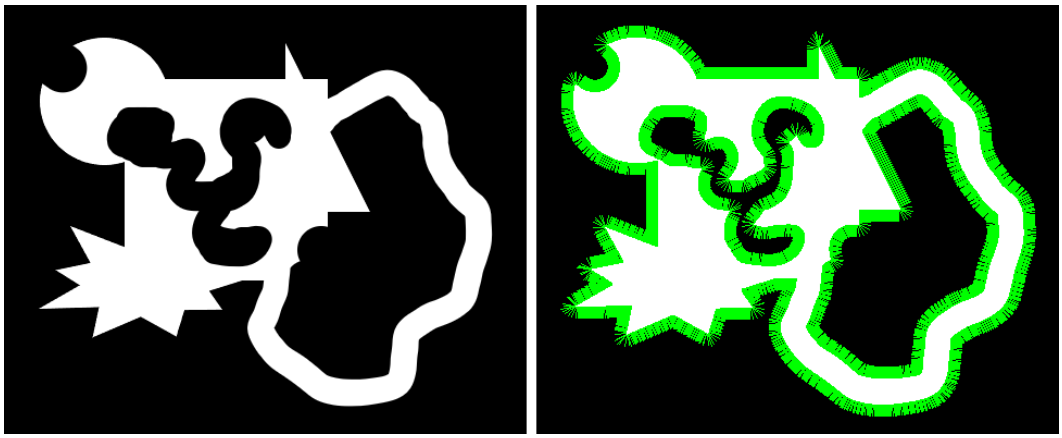


Figure 3.9 The normal extraction result (right) of a random shape (left). The white part is the silhouette of a complex geometry, and the green lines are the calculated normal vectors.

3.2.2 High-Level Quality Features

Based on the low-level features, a set of more in-depth features that can be used to quantify the quality of a grasp pose was created. These high-level features are the missing indicator, the collision indicator, the object translation predictor, the object rotation predictor, and the type-I and

type-II object slippage predictors. The missing and collision indicators are extracted in the process of finding the low-level visual features. This section explains the calculation processes of the other high-level quality features.

The *object translation predictor (D_t)* estimates the ratio of the object travel distance (T_d) during gripper closing to half of the gripper's full-open width ($r/2$). As shown in Figure 3.10, the object translation distance T_d is measured from the center of the two primary contact points to the grasp rectangle's center. The grasp distances of the left and right primary contact points are the minimum and maximum grasp distances in the $GRPV^L$ and $GRPV^R$, respectively. The ratio of the object translation distance to the gripper's half-open width can be calculated using equation (3.14). For this translation feature, we use the distance ratio instead of the distance as the grasp quality measure because using the ratio allows this feature to adapt to objects and grippers of different sizes.

$$D_t = \left| \frac{min(GRPV^L) + max(GRPV^R)}{r} \right| \tag{3.14}$$

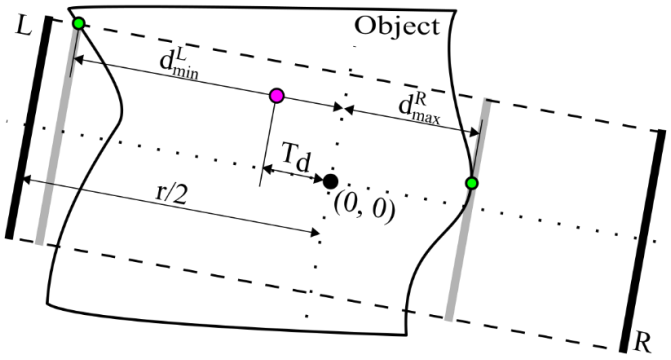


Figure 3.10 The object translation predictor illustration diagram.

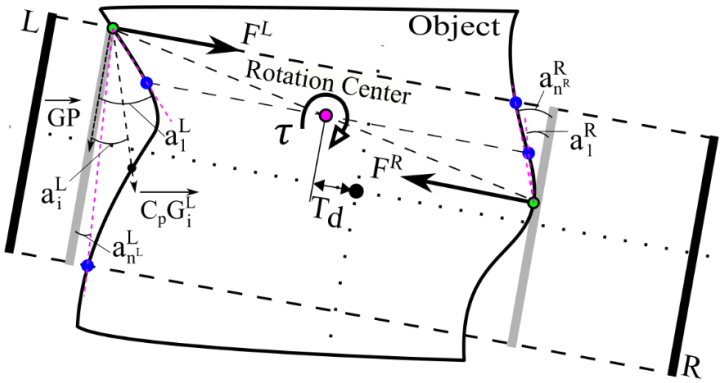


Figure 3.11 The object rotation predictor illustration diagram.

The *object rotation predictor* (R_r) predicts the angle of object rotation during grasping. The object rotates when there is torque, and the torque occurs when the two primary contact points are not on the same grasp line. With the two primary contact points, the rotation center and the rotation direction can be determined. The *potential after-rotation contact region (PARCR)* is defined using the center of rotation and the rotation direction. As shown in Figure 3.11, the PARCR on each side is the object outline segment between the two grasp points highlighted in blue. One grasp point is on the same grasp line as the rotation center, and the other is the last grasp point of the region approaching the gripper in the rotation process. Once the PARCR's are determined, we calculate the rotation angle by selecting the minimum required rotation for a grasp point in the PARCR to be the new contact point after the rotation. Geometrically, the rotation angle (a_i) is defined as the angle between the gripper plate and the line that connects the primary contact point (C_p, the approximate pivot point) and the new after-rotation contact point. The mathematical expression of this rotation feature is shown in equation (3.15)

$$R_r = min\left(a_1^L, a_2^L, \cdots, a_{n^L}^L, a_1^R, a_2^R, \cdots, a_{n^R}^R\right) \tag{3.15}$$

where a_i^e's are the rotation angles. Each rotation angle is calculated as the angle between the gripper plate vector $\overrightarrow{GP}$ and $\overrightarrow{C_pG_i^e}$, the vector from the primary contact point to the i^{th} grasp point in the PARCR.

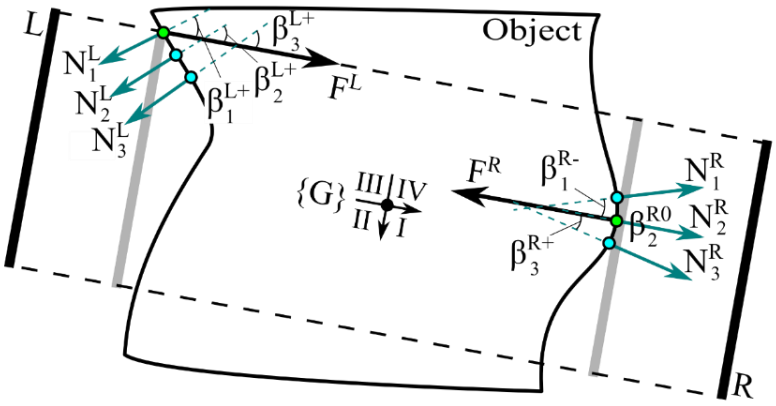


Figure 3.12 The type-I slippage predictor illustration diagram.

The *type-I slippage predictor* (S_a) is a feature that measures how likely the object will slip during grasping due to the slope of the contact surfaces. This feature is defined as the average *contact angle* of the most slippery contact region. As shown in Figure 3.12 and equation (3.16), the contact angle ($\beta_i^e \epsilon [-90, 90]$) is defined as the angle between the contact force (F) and the contact point's inward normal (-N_i). If the contact point normal (N_i) points towards the first and second quadrant of the projected-gripper frame {G}, then the contact angle is positive; and it is negative otherwise.

$$\beta_i^e = \begin{cases} \angle(F^e, -N_i^e), & N_{iy}^e > 0 \\ -\angle(F^e, -N_i^e), & N_{iy}^e < 0 \end{cases} \tag{3.16}$$

where N_{iy}^{e} is the y coordinate of the normal N_i^e in the projected-gripper frame {G}. After getting the contact angles, we derived a slippage indicator SI^e to indicate the chance of the type-I slippage occurring. This slippage indicator is calculated by equation (3.17).

$$SI^e = \sum_{i=1}^{n^e} |\beta_i^e| - \left| \sum_{i=1}^{n^e} \beta_i^e \right| \tag{3.17}$$

where n^e is the number of contact points of the *e* side of the gripper. When SI^e equals 0, all contact angles of the corresponding side must have the same polarity. If one contact region has the same polarity contact angles, then the contact region is slippery, and the slipperiness is indicated by the average of all the contact angles. The larger the average contact angle, the more likely the object will slip. Therefore, the type-I slippage predictor is calculated as a scale of how likely the slippage will occur during gripper closing, as shown in equation (3.18).

$$S_a = \begin{cases} \frac{1}{2}\left(\left| \frac{\sum_{i=1}^{n^L} \beta_i^L}{n^L} \right| + \left| \frac{\sum_{i=1}^{n^R} \beta_i^R}{n^R} \right| \right), & SI^e = 0 \\ 0, & SI^e \neq 0 \end{cases} \tag{3.18}$$

The *type-II slippage predictor (S_f)* is a feature that predicts slippage through the grasp force placement. This feature favors grasp forces that are balanced and centered on the gripper contact surface. It is calculated as the average of the left and right force zone center offsets using equations (3.19) and (3.20).

$$S_f = \frac{S_f^L + S_f^R}{2} \tag{3.19}$$

$$S_f^e = \frac{min(CPD^e) + max(CPD^e)}{4} + \frac{C_g^e}{2} \tag{3.20}$$

where $min(CPD^e)$ and $max(CPD^e)$ are the minimum and maximum contact point center offset of the side *e (left or right)*, they are Δw_{min}^e and Δw_{max}^e (equation 3.3), respectively; C_g^e is the center of the grasp region of side *e*. As shown in Figure 3.13, the gray region between two contact points (green squares) is the force zone of the corresponding side of the gripper. The force zone contains all primary and secondary contact points. For each side of the gripper, the force zone center offset (dark green circle) is the average of the force zone center (orange circle) with the grasp region center (purple circle).

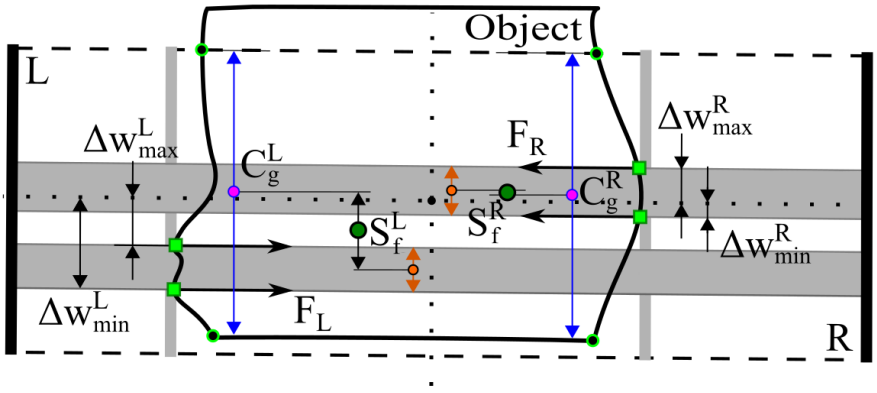


Figure 3.13 The type-II slippage predictor illustration diagram. Gray region: contact force region, orange circle: contact force region center, pink circle: grasp region center, dark green circle: force zone center offset.

Note that, in this work, the friction force was not directly considered when predicting slippage for two reasons. First, directly calculating friction force requires knowledge of contact forces and friction coefficients, which is nearly impossible to obtain from only visual input. Also, the goal is to predict grasp quality before the robot executes it, so there may never be tactile input.

Second, assuming the robot gripper can exert enough force to pick up the object when the grasp pose is appropriate is reasonable. Under this assumption, a relative measure of the slippage likelihood can be developed from the contact forces' locations and angles while ignoring the magnitudes of the forces. Minimizing the slippage predictors is optimizing the conditions of the contact forces applied to the object to achieve the most balanced grasp conditions, which will automatically maximize the friction forces applied to the object.

3.3 Grasp Quality Scoring

The quality features are in different units and scales. Before combining the quality features, each feature is first normalized by its linear normalization function defined by two endpoints (0, 1) and (τ_i, 0). The endpoints' x-coordinates are the feature values, and the y-coordinates are the normalized feature scores. The τ_i is the threshold feature value when the feature score equals 0. Since the translation predictor D_t is a ratio, its zero-score threshold value is 1. The rotation predictor R_r and type-I slippage predictor S_a are measures of angles, and their thresholds are set to 60 degrees based on experiments. The type-II slippage predictor S_f is the force placement feature, and its value ranges from 0 to the gripper projection's half-width (w/2). Two feature thresholds, w/4 and w/2 are used for feature scores 0.75 and 0, respectively, which makes the function have two different slopes when normalizing low and high feature values. All the parameters in the normalization functions are empirically determined to match the real grasping result.

After feature normalization, the grasp quality measure S can be calculated as the weighted sum of the feature scores using equation (3.21).

$$S = ks_{min} + (1 - k)s_o \quad (3.21)$$

where s_{min} is the minimum feature score, and s_{o^-} is the average of other feature scores. Since the grasp pose's quality mainly depends on its worst quality feature, the weight *k* should be assigned in a minimum-dominant fashion ($k >> 0.5$). In this work, *k* is set to 0.9, which was empirically determined by specifying the grasp quality to the desired value when $s_{min} = 0$ and $s_{o^-} = 1$.

3.4 Computational Cost

The computation process of the presented grasp quality measure includes the extraction, normalization, and combination of the grasp features. The most time-consuming part is the grasp feature extraction process. The main parameters that affect the computational time are the projected gripper opening width and the number of grasp lines. The gripper opening width determines the search space size for finding the grasp point. The number of grasp lines determines how many rounds of grasp point searching need to be performed and the input size in high-level feature computation.

Figure 3.14 shows the average computational time for different gripper opening widths and the number of grasp lines. These results were acquired by averaging the computational time of 1000 grasp quality evaluations for each combination of gripper opening width and grasp line count. Based on these results, we can see that the computational cost of the presented grasp quality measure has a linear relationship with the number of grasp lines. Because of the step size optimization (equation (3.6)) in grasp point searching, the increase of the computational cost is less than a linear rate when the gripper opening width increases. In actual applications, one can easily adjust the computational cost by manipulating the scale of the gripper and object projection image to make the projected gripper have a desired opening width and grasp line count.

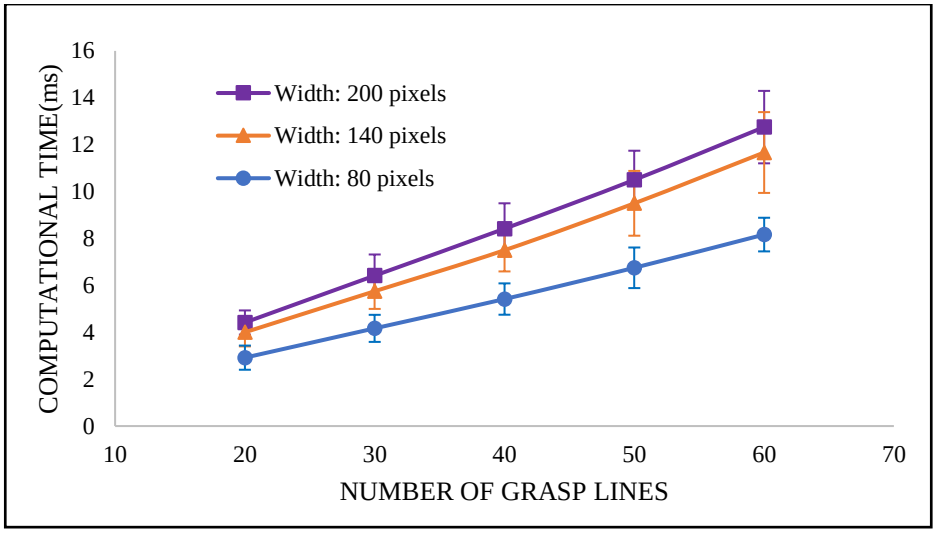


Figure 3.14 The computational cost of the grasp quality measure for different combinations of gripper opening widths and grasp line numbers.

Chapter 4: Robot Vision Module

The robot vision module is the foundation of the developed HitL robot grasping system, as it provides visual information about the scene and object for the other modules of the system. The key functions of the vision module are 3D visual data perception and object recognition and segmentation. Object recognition and segmentation are achieved using the user's selection and the raw scene image from the camera as input, and outputs the name of the user-selected object with a mask image that segments the object from the background. The 3D perception is performed by combining the scene point clouds from two views to obtain the 3D geometry information of the scene and the object to grasp. More accurate object geometry information can be obtained for known objects by matching the object point cloud in a database to the scene point cloud. Generally, this vision system takes camera data from two views as input and only needs one view if the object is known with its point cloud model saved in the database. The vision module was built upon existing open-source works. Our contribution in this part is optimizing and integrating the existing works to form the vision module that meets the needs of our assistive robots.

4.1 Camera to Robot External Calibration

The camera's pose relative to the robot must be properly calibrated to ensure the accuracy of the vision system. In this dissertation, the calibration method depends on the robot's repeatability instead of accuracy. This type of method is preferred because robots often have much better repeatability than accuracy. The calibration process is to use the robot gripper to measure the pose of the calibration pattern (${}^{B}T_{P}$) first, then use the camera to detect the calibration pattern and calculate its pose (${}^{C}T_{P}$) in the camera frame. Lastly, the camera's pose in the robot base frame

(${}^{B}T_{C}$) can be calculated using equation (4.1). For the hand camera, because it is fixed to the robot's end-effector link, its external pose should be calibrated relative to the robot gripper frame (${}^{G}T_{C}$) using equations (4.1) and (4.2). For other camera placements, the calibration goal should change accordingly to use a frame with a fixed transformation to the camera frame as the reference.

$$ {}^{B}T_{C} = {}^{B}T_{P} \ {}^{C}T_{P}{}^{-1} \tag{4.1} $$

$$ {}^{G}T_{C} = {}^{B}T_{G}{}^{-1} \, {}^{B}T_{C} \tag{4.2} $$

where the letters B, C, P, and G refer to the robot base frame, the camera frame, the calibration pattern frame, and the robot gripper frame, respectively. ${}^{B}T_{G}$ is the transformation matrix of the gripper frame in reference to the robot base frame, which is obtained from the robot's forward kinematics.

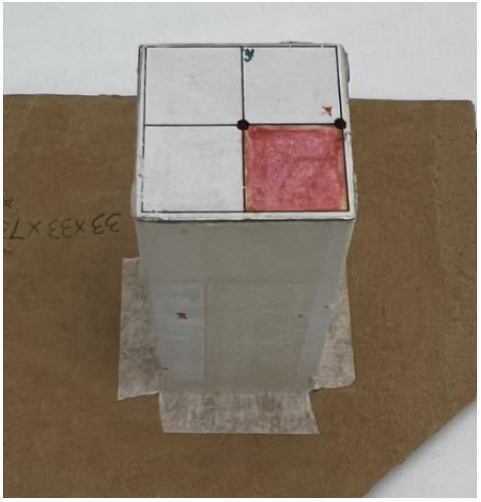


Figure 4.1 The calibration pattern used in the camera to robot calibration.

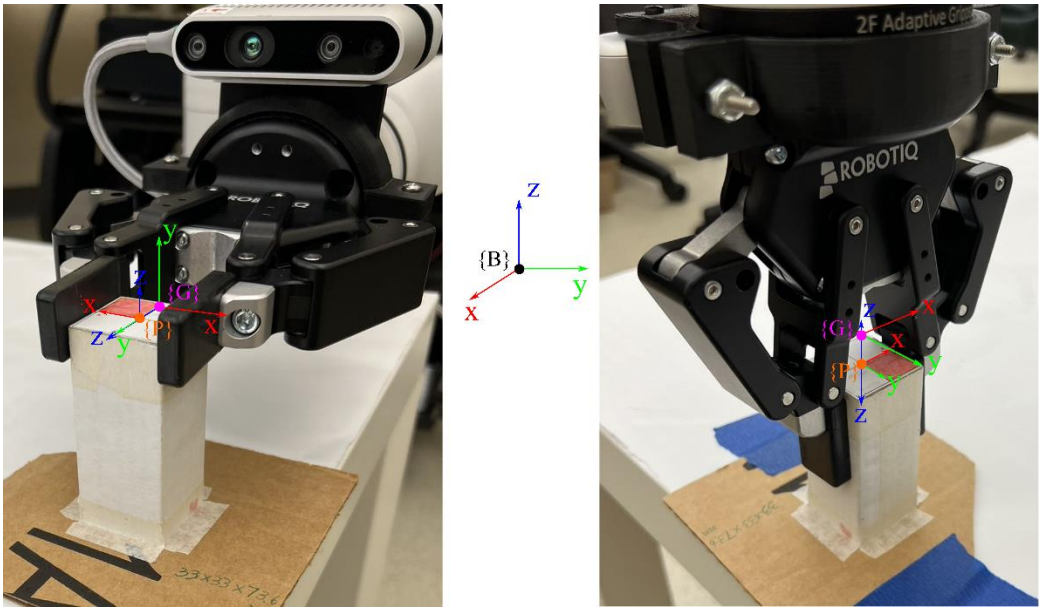


Figure 4.2 Illustration of robot placing and measuring the pose of the calibration pattern. The letters B, G, and P refer to the robot base frame, the gripper frame, and the calibration pattern frame.

The used calibration pattern is a simple grid pattern attached to a cuboid supporting base, as shown in Figure 4.1. The grid pattern defines the x and y axes of the pattern frame, and the supporting base is used to help align the pattern with the robot's gripper frame. As illustrated by Figure 4.2, to use the robot gripper to measure the pose of the calibration pattern, the first step is to orient the robot gripper in the horizontal direction where the gripper's x-z plane is parallel to the robot base's x-y plane. Then the height of the gripper is adjusted to align the x-z plane of the gripper frame to the calibration pattern's x-y plane. After this process, the z-coordinate of the calibration pattern is the same as the z-coordinate of the robot gripper. After the z-coordinate of the pattern is measured, the next step is to move the robot gripper to a vertical pose where the z-axes of the gripper and the robot base are in opposite directions, and the gripper's x-z plane is parallel to the y-z plane of the base. Because the support of the calibration pattern is a cuboid, with proper initial placement, the calibration pattern's orientation will be forced to align with the robot

base when it is grasped by the gripper using the pre-defined vertical pose. In addition, the position of the calibration pattern can be adjusted to align with the center of the gripper so that the calibration pattern's x and y coordinates are the same as the ones of the gripper. Through this process, the pose of the calibration pattern in the robot base frame is determined as in equation (4.3).

$$ {}^{B}T_{P} = \begin{bmatrix} 1 & 0 & 0 & x_g \\ 0 & 1 & 0 & y_g \\ 0 & 0 & 1 & z_g \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3} $$

where x_g, y_g, and z_g are the robot gripper's positional coordinates.

After the calibration pattern pose is measured relative to the robot base frame, its pose relative to the camera frame (${}^{C}T_{P}$) can be calculated using a set of sample points on the pattern surface in the camera view. For the best accuracy, at least two sample points should be manually selected from the pattern image. The first point is the center point of the pattern, which represents the position of the pattern; the second point must be a point on the pattern's x-axis; this helps the algorithm determine the orientation of the pattern. Besides the two main sample points, 10 to 20 more sample points across the pattern surface should be selected from the image. The plane equation of the pattern surface can be calculated using the selected sample points. The pattern frame's z-axis is the normal of the pattern surface plane.

To calculate the plane equation of the calibration pattern surface is to calculate the plane equation parameters (p) such that l_z is minimized. l_z is the L^2 norm of the z-distances of the sample points to the pattern surface plane, which is computed using equations (4.4) to (4.7).

$$ l_z = \left\| v_z - M_{xy\_}\, p \right\|_2 \tag{4.4} $$

$$v_z = [z_1, z_2, z_3, \dots, z_n]^T \tag{4.5}$$

$$M_{xy\_} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \tag{4.6}$$

$$p = [a, b, c]^T \tag{4.7}$$

$$z = ax + by + c \tag{4.8}$$

In equations (4.5) and (4.6), x_i, y_i, and z_i (i =1, 2, 3, … , n) are the coordinates of the sampled surface points of the calibration pattern. Equation (4.8) is the plane equation based on the parameters in equation (4.7). The vector of the plane parameters is computed as the least-square solution of the equation $v_z = M_{xy\_}\, p$. The least square is computed using the SciPy [49] library.

4.2 Object Recognition and Segmentation

Object recognition and segmentation are achieved using the Detectron2 [38] object recognition and segmentation neural network. The Detectron2 library provides state-of-the-art detection and segmentation algorithms developed by Facebook AI Research. Figures 4.3 and 4.4 show object recognition and segmentation examples using this neural network. The network generally works very well, but it is not stable enough for a reliable robot vision system. The instability is reflected in two aspects. One is that the network is sensitive to the view angle in recognizing objects. As shown in figure 4.3, frames 1 and 2 are images of the same objects from different view angles. The tennis ball (recognized as an apple) and the toy lemon (recognized as orange) are recognized in frame 1 but not in frame 2. Another aspect of the instability is that the recognition results could differ from one frame of the image to another for the same object. As

shown in Figure 4.4, the network recognizes the tennis ball correctly in frame 1 as a tennis ball, but in frame 2, it recognizes the tennis ball as an apple, and in frame 3, it does not recognize the tennis ball at all.

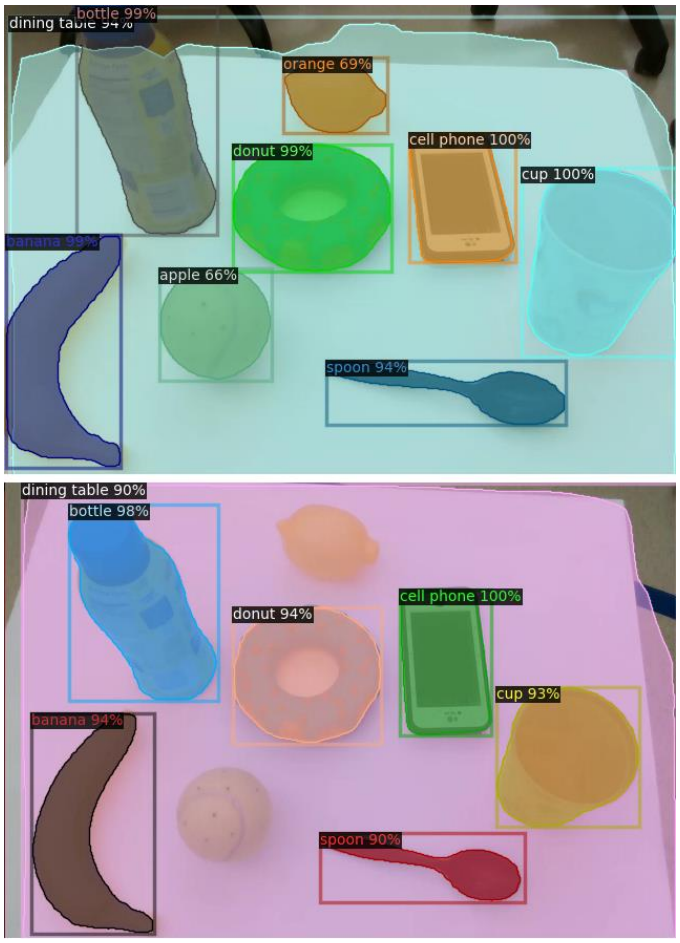


Figure 4.3 Object recognition results of different frames from different view poses.

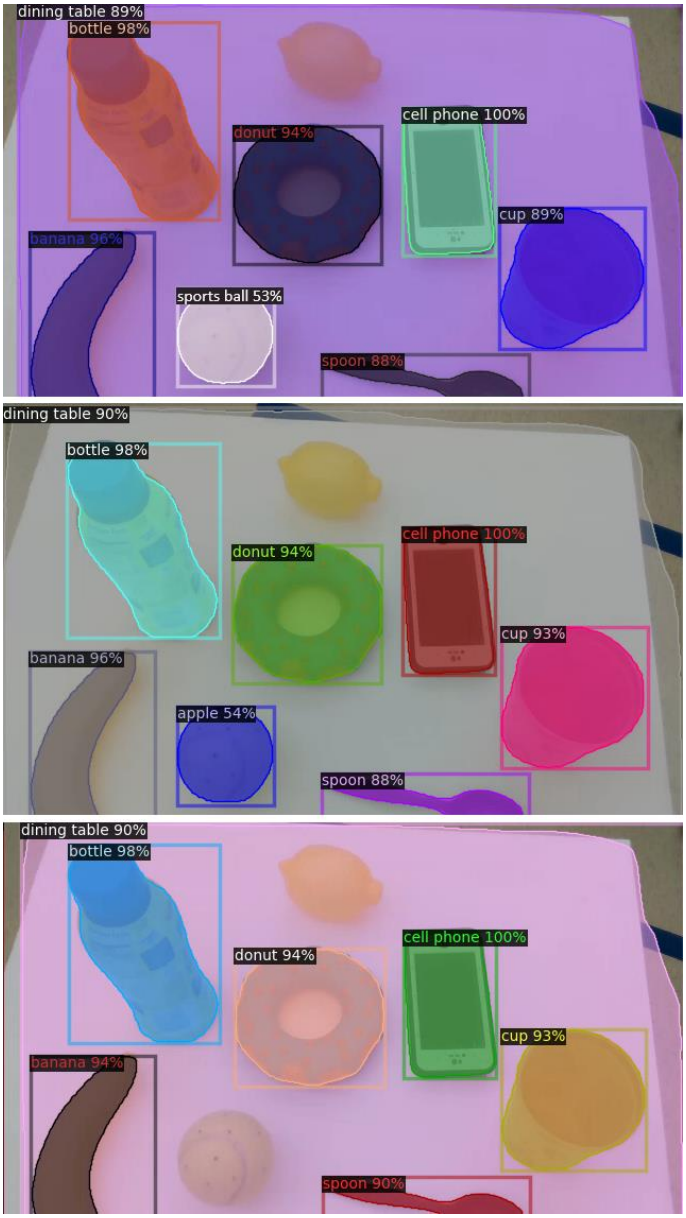


Figure 4.4 Object recognition results of the same view images from different times.

The object recognition process is optimized to make the object recognition and segmentation network fit for the robot vision module. The first optimization uses the user input and a graspable object list to filter the output so that the robot only receives information about the object of interest from the vision module. As shown in Figures 4.3 and 4.4, the raw output of the network contains the name, the recognition confidence score, and the segmentation mask of all

recognized objects in the image. Ungraspable objects, such as the dining table and chair, are removed from the raw recognition output by checking if the detected object is on the graspable object list. The pixel location selected by the user from a graphical user interface is used to find the closest graspable object within a preset radius. Figure 4.5 shows the object recognition and segmentation result of an object of the user's interest after output filtering.

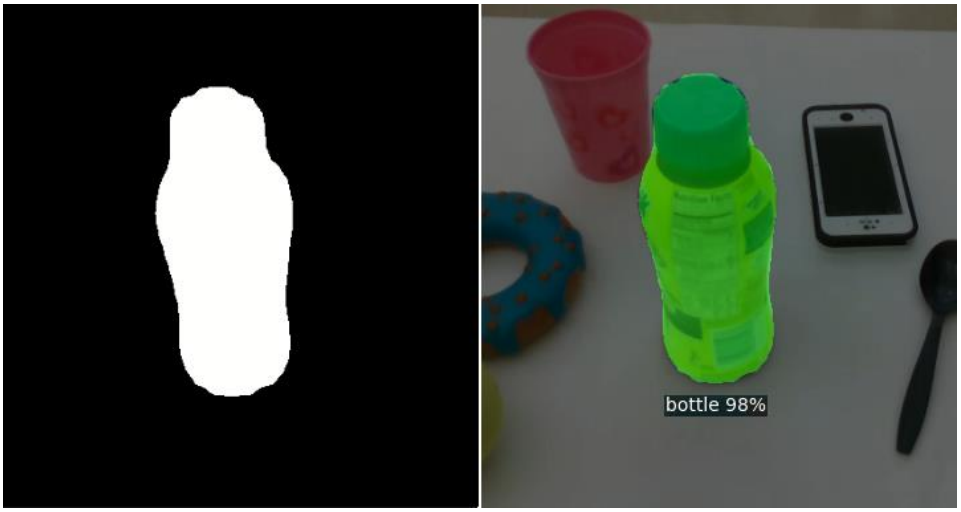


Figure 4.5 Object recognition and segmentation example result of a user-selected object. Left: object segmentation mask, right: original RGB image with the object segmentation mask, the object name, and recognition confidence.

The second optimization is stabilizing the output of the object recognition network using batch input and result voting. Batch input is to use multiple consecutive frames as input for object recognition and segmentation. Result voting assigns votes to the object recognition result of each frame in the input batch. Once the object recognition result of each frame is acquired, the votes for a result from one frame are calculated as the base votes times the recognition confidence score. For the ones that have the same recognition result, their votes are added up. The final recognition result is the result that has the most votes.

In this dissertation, five frames were used in one input batch, and each frame had 100 base votes. For example, let the tennis ball recognition results of 5 input frames be: sports ball with 87% confidence, sports ball with 73% confidence, apple with 65% confidence, no recognition result, and sports ball with 60% confidence, respectively. In this case, frames 1, 2, and 5 contribute 220 (87+73+60) votes for sports ball, frame 3 contributes 65 votes to apple, and frame 4 contributes 100 votes to unknown object. Therefore, the final recognition result will be "sports ball with 220/500 votes". Using batch input and result voting prevents the object recognition result from being affected by network blips, increasing the stability of the object recognition and segmentation.

With these optimizations, the stability of the object recognition and segmentation module is sufficient. To further increase the module's reliability, the problems related to recognition failure need to be appropriately handled. Recognition failure can be of two types: when the network recognizes the object as the wrong object and when the network does not recognize a recognizable object. The first type of failure is inevitable because, in a real-life environment, objects will always look like other objects to the vision system based on current technology. For example, the tennis ball is often recognized as an apple, as shown in Figures 4.3 and 4.4. As there is no fix to the first type of failure, the only strategy that can be used to increase reliability is to identify the failure and not use the recognition results in the downstream processes. The second type of failure is often caused by a bad view angle, as shown in Figure 4.3. This failure can be addressed by changing the view angle to the direction where the object can be recognized. Based on how the recognition failure should be handled and our robot grasping system always has a human user in the loop, it makes perfect sense to transfer the problem to the user control loop so that the recognition failure problem can be handled by human intelligence. If the vision system outputs false recognition results for grasping known objects, the user can identify the recognition failures and switch the

process to unknown object grasping. If the object is not recognized because of a bad view angle, the user can use the assisted velocity control module (Chapter 7) to adjust the view angle.

4.3 3D Visual Perception

The main function of the 3D visual perception module is retrieving the object's 3D geometry and 6D pose information. The point clouds are pre-scanned offline and saved in a database for known objects. When the object recognition and segmentation module recognizes the object, the object's point cloud model can be retrieved from the object database using the recognized object name. The object's 6D pose can be obtained by matching the object model to the object's partial point cloud in the live scene. In this work, we used OpenCV's point pair features (PPF) surface matching algorithm [30] to perform the object model-to-scene matching. The block diagram for known object perception is shown in Figure 4.6.

The PPF surface matching algorithm [31] is developed based on the point pair features (PPF). The PPF for a pair of points include the distance between the two points, the angle between the two points' normals, and the angles between the two points' difference vector and their respective normals (a visualization of the PPF definition can be found in [28]). The performance of this surface-matching algorithm depends on the number and the normals of the points to be matched. The surface matching process has two stages. The first stage is extracting and indexing PPF from the object model. The second stage is extracting PPF from the scene point cloud and comparing them to the object model's PPF. From the comparison, the rough object-to-scene match pose can be found using Hough-like voting and clustering [31].

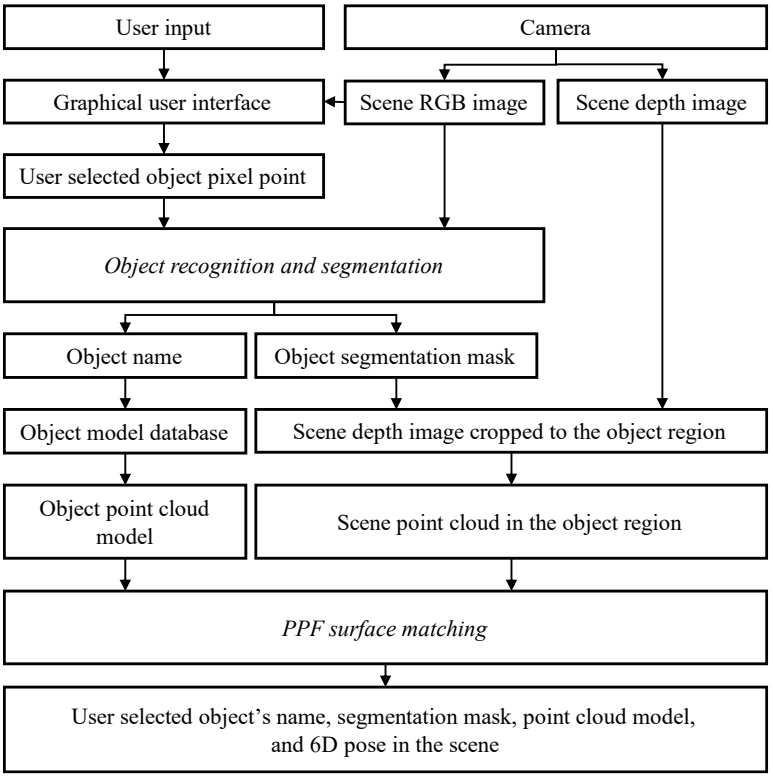


Figure 4.6 Block diagram of the robot vision system for known objects.

When implementing the PPF surface matching algorithm, two main tweaks are implemented to achieve optimal performance. The first tweak is downsampling both the object model and the scene point cloud to reduce the computational time. The rules for determining the downsampling ratio were: (1) The computational time of the matching process should be less than 0.1 seconds. (2) There must be enough points in the point cloud of the model and scene to preserve their actual geometry. For the scene point cloud, only down sampling is not enough because the PPF are local features, and many places in the scene could share the same local features. The scene point cloud must be cropped to the region of the object to minimize the chance of mismatching. As it is difficult to find the object's bounding box in 3D space to crop the point cloud of the scene, an alternative method is to crop the scene depth image to the object region and calculate the object point cloud from the cropped depth image. The bounding box for cropping the depth image can be

obtained from the object recognition and segmentation result. Figure 4.7 shows an example of the object bounding box and the object point cloud calculated from the cropped scene depth image.

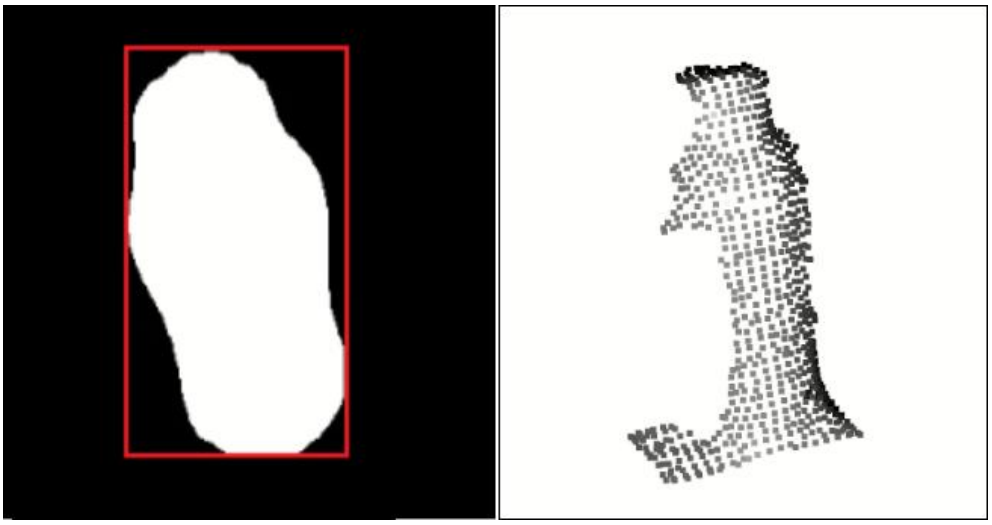


Figure 4.7 Illustration of the scene depth image's crop region (red rectangle on the left) and the cropped scene point cloud (right).

The second tweak is to use batch input and result voting again to increase the stability of the matching results. This tweak uses the same method described in Section 4.1. The only difference is that the PPF surface matching result, in this case, includes a vote measurement that can be used directly. Figure 4.8 shows an example of the object model matched to the scene using this 3D visual perception module. The green points are the object point cloud model obtained from the object database, and the gray points are the scene points in the object region. This visualization of the matching result will be displayed in the user interface so that, when false matching occurs, the human user can identify the error and switch the system to the unknown object grasping mode.

The 3D geometry information is retrieved for unknown objects by combining the point clouds from the robot hand camera in different view directions. The object point cloud can be found in the scene point cloud based on the user-selected point and the robot's gripper pose

generated by the grasp detection module. The point cloud around the user-selected point and within the gripper's grasp region is the point cloud of the object of interest. When the camera is in a view pose to get the scene point cloud, the camera's pose in the robot base frame should also be calculated so that the received point cloud can be transformed into the robot base frame for point cloud combination. The accuracy of the obtained unknown object geometry information mainly depends on the accuracy of the camera sensors and the camera-robot calibration. Figure 4.9 shows an example of the combined scene point cloud from two views.

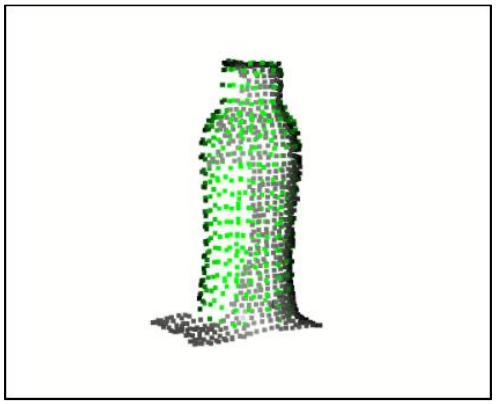


Figure 4.8 Example of object model matching using the 3D visual perception module. Green points: object point cloud model, gray points: cropped scene point cloud.

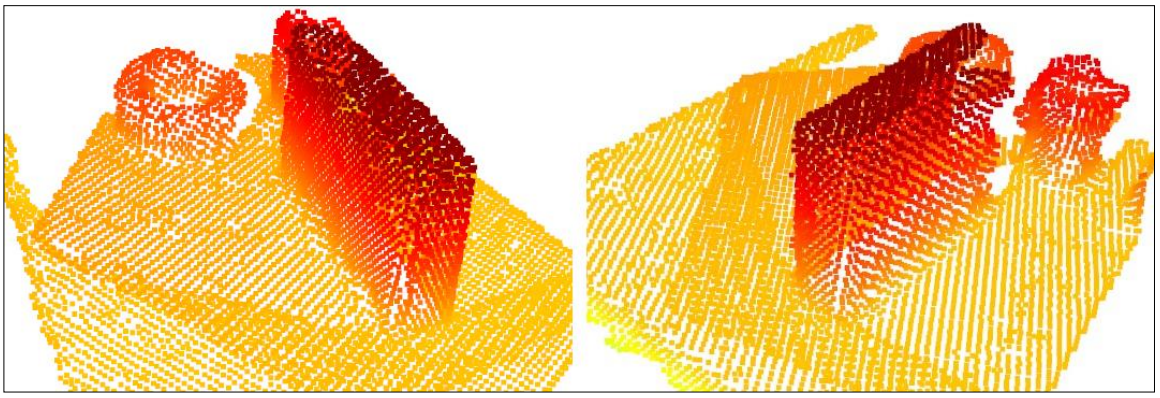


Figure 4.9 An example of the combined scene point cloud from two views. The color scale is based on the point's depth.

Chapter 5: Deep Learning-Based Grasp Detection Module

The deep learning-based grasp detection module was developed based on a grasp pose detection network and a grasp pose filtering technique. The grasp detection network generates dozens of grasp poses for the user-selected object. The grasp filtering system first categorizes the grasp poses into groups based on the grasp poses' grasp directions, then filters the grasp poses in each category. Figure 5.1 shows a block diagram of the grasp detection module. Our contribution in this chapter is developing the grasp filtering technique to help select the final grasp pose from the outputs of the grasp detection network.

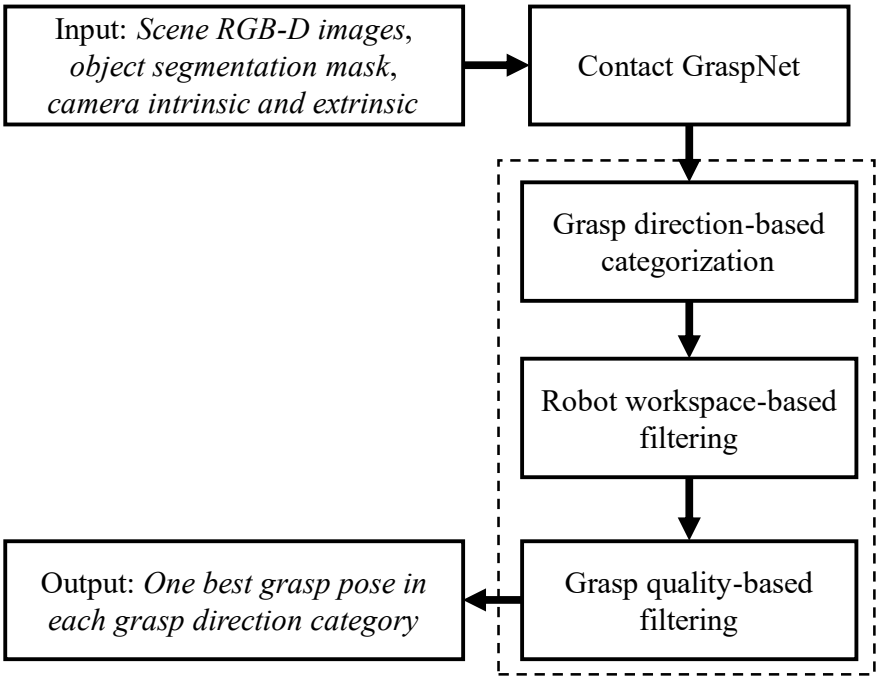


Figure 5.1 Block diagram of the deep learning-based grasp detection module.

5.1 Grasp Pose Detection

This module uses the Contact-GraspNet (CGN), a state-of-the-art 7D grasp (6D pose and the gripper's open-width) detection neural network [25], as the backbone. The default output of the CGN contains many possible grasp poses for all graspable objects in the scene. If we apply the object's full segmentation mask, as shown in the top row of Figure 5.2, the detected grasp poses can still be abundant. Therefore, we use a small square mask centered at the user-selected point for grasp detection, which allows us to obtain grasp poses only around the user-selected point, as shown in the bottom row of Figure 5.2.

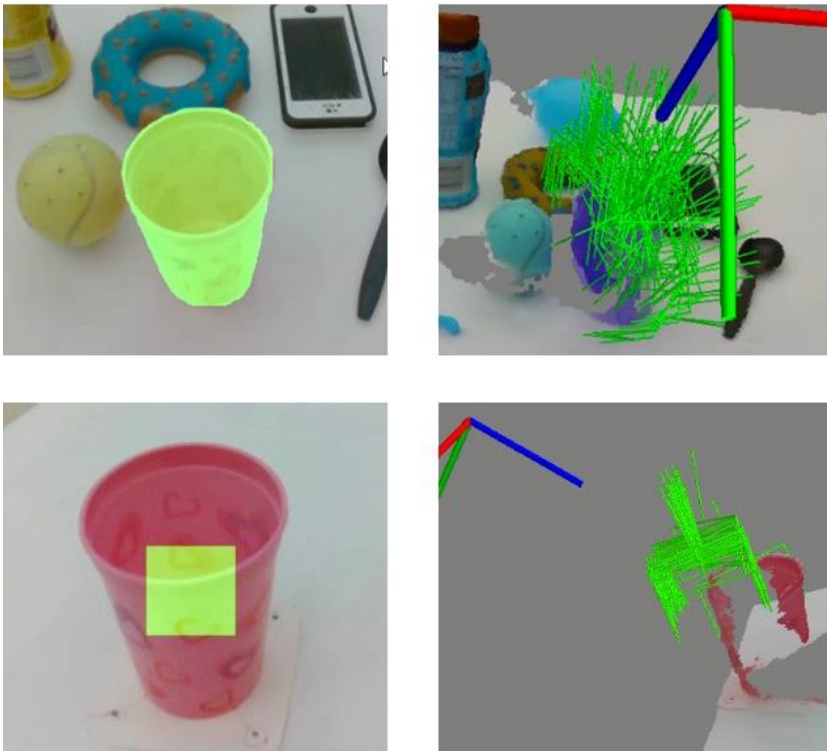


Figure 5.2 Examples of grasp detection based on different object masks. The top and bottom rows show the detected grasp poses using the cup's full segmentation mask and the small square mask at the user-selected point. The masks are colored green, and the grasp poses are shown as green lines.

5.2 Grasp Pose Filtering

After the grasp detection network generates the grasp poses, the next challenge is to select an appropriate grasp pose from the many possible options. By default, the best-quality grasp pose is selected based on the quality measure generated by the grasp detection network. However, in real robot applications, the final grasp pose should also meet the criteria that include different aspects of the task, such as collision avoidance, the robot workspace limits, the post-grasp task requirements, and user preference. Therefore, the grasp pose filtering system is developed to improve the final grasp selection based on different properties of the grasp. In the following list, the most important properties that affect the grasp pose selection are discussed:

1. The grasp pose's grasp direction. For many objects, many grasp poses in different grasp directions can be possible if we only consider the objects' geometric properties. However, in real robot applications, the user's selection and the post-grasp task requirements must also be considered. Therefore, the grasp directions should be extracted to provide more information about the grasp poses to facilitate the selection of the optimal pose. For example, grasping a cup full of liquid must be kept level.
2. The robot's workspace limitation. Because the grasp detection is based only on the object depth data, there could be grasp poses suitable for the object but not reachable by the robot. Checking if the grasp poses are within the robot's workspace can find and remove the unreachable poses.
3. Possible collisions. Before the robot performs a grasp, it is essential to check if the grasp will cause any collision. However, this is not considered in this filtering system for two reasons. The first one is that the grasp detection network is quite reliable in generating collision-free grasp poses; therefore, the chance of the detected grasp pose causing any collision is low. The second reason is that even if the selected grasp pose would cause any collision, it cannot pass

the grasp refinement module, and the grasp refinement process can adjust the grasp pose to avoid the collision. Therefore, this case is only listed here for the completeness of the properties that could affect the grasp pose selection.

4. The grasp location relative to the object. Again, because the grasp detection network does not differentiate different parts of the object, grasp poses will be detected at all geometrically graspable parts. While in real robot grasping, the part of the object that should be grasped often depends on the post-grasp task. For example, when grasping a knife to be used for cutting, the robot should grasp the knife by the handle; but when grasping the knife to pass it to a human, the robot should grasp the knife by the blade. In this dissertation, this property is not implemented in the filtering process, as task-based grasping is outside the scope of the current work. This property is listed here for the completeness of the properties that could affect the grasp pose selection.

Based on the listed properties, the grasp pose filtering is developed based on the grasp poses' grasp directions and the robot's workspace limitations. In the first filtering stage, the grasp poses are categorized based on their grasp directions. In the second filtering stage, the grasp poses in each grasp direction category are filtered based on the robot workspace limits and the grasp pose quality score. After this filtering process, the best grasp poses in different grasp directions are saved for later selection based on the user's preference or the task requirement; the grasp poses outside the robot workspace or do not meet the quality requirement are removed.

An object frame {O} is assigned to the object, as shown in Figure 5.3, to help categorize the grasp poses. The object frame is the robot base frame {B} translated to the user-selected object point. The grasp direction of a grasp pose in the object frame can be represented by a vector in Cartesian coordinates $(x_d,\ y_d,\ z_d)$ or in spherical coordinates (θ, φ, r). The grasp direction can be

defined by two parameters using the spherical coordinates, the zenith (θ) and azimuth (φ) angles. The grasp direction categories are defined in Table 5.1, and figure 5.4 illustrates the grasp direction categories with respect to the zenith and azimuth angles separately.

Table 5.1 The definition of the grasp direction categories based on the zenith and azimuth angles.

Parameter	Category				
θ	Bottom-up $[0^{\circ}, 10^{\circ})$	Angled-up $[10^{\circ}, 80^{\circ})$	Side $[80^{\circ}, 100^{\circ})$	Angled-down $[100^{\circ}, 170^{\circ})$	Top-down $[170^{\circ}, 180^{\circ}]$
φ	I $[-90^{\circ}, -54^{\circ})$	II $[-54^{\circ}, -18^{\circ})$	III $[-18^{\circ}, 18^{\circ})$	IV $[18^{\circ}, 54^{\circ})$	V $[54^{\circ}, 90^{\circ}]$

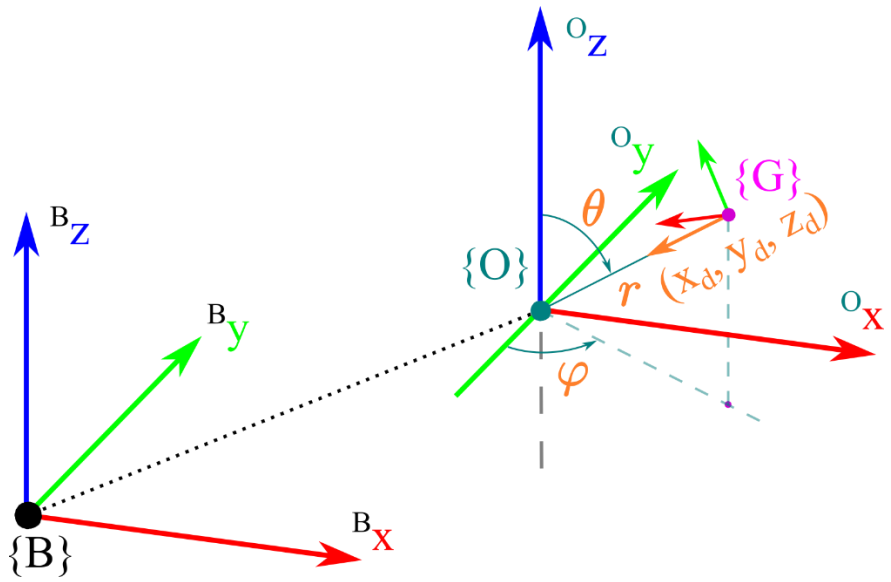


Figure 5.3 Illustration of the grasp direction vector (x_d, y_d, z_d) in the object frame and the object frame {O} relative to the robot base frame {B}. The grasp direction vector, colored orange, is the z-axis of the gripper frame {G}.

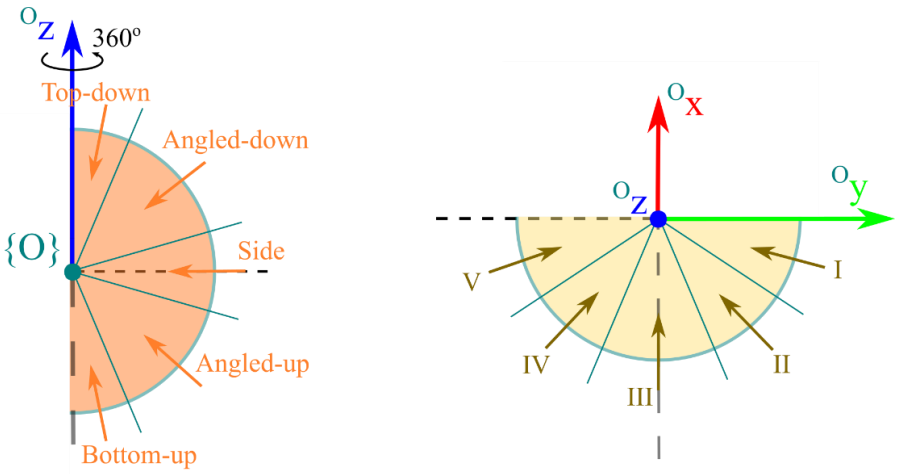


Figure 5.4 Illustration of the grasp direction categories. Left: zenith angle-based categories; right: azimuth angle-based categories.

The grasp direction categories defined by the zenith angle θ are the main categories, and the ones defined by the azimuth angle φ are the sub-categories. There are five more sub-categories for each of the angled-down, side, and angled-up main categories. For the top-down and bottom-up categories, having sub-categories is unnecessary because the grasp direction is close to the singularity, where φ has little to no effect on the grasp direction.

Using Cartesian coordinates-based criteria for categorizing the grasp pose is more convenient for coding. Therefore, utilizing the conversion equations (5.1) to (5.3), the grasp direction categories defined by the spherical coordinates (as shown in Table 5.1) can be converted to a Cartesian coordinates-based categorization, as shown in Table 5.2. The radial distance r is equal to 1 for a unit direction vector.

$$x_d = r \sin\theta \cos\varphi \tag{5.1}$$

$$y_d = r \sin\theta \sin\varphi \tag{5.2}$$

$$z_d = r\cos\theta \tag{5.3}$$

Table 5.2 The definition of the grasp direction categories based on the Cartesian coordinates of the grasp direction vector.

Parameter	Category				
z_d	Bottom-up $[1, 0.985)$	Angled-up $[0.985, 0.174)$	Side $[0.174, -0.174)$	Angled-down $[-0.174, -0.985)$	Top-down $[-0.985, -1]$
y_d/x_d	I $(-\infty, -1.376)$	II $[-1.376, -0.325)$	III $[-0.325, 0.325)$	IV $[0.325, 1.376)$	V $[1.376, \infty)$

Before categorization, the grasp poses must be transformed to the robot base frame using equation (5.4). Because the object frame and the robot base frame have the same orientation, a grasp pose's direction vector in the object frame is the same as its direction vector in the robot base frame. In addition, the direction vector of a grasp pose is a unit vector in the direction of the gripper frame's z-axis. Therefore, the grasp direction vectors of the grasp detection network's output grasp poses can be calculated using equations (5.4) and (5.5). The direction categories can be found by comparing the grasp direction vectors to Table 5.2.

$$^{B}T_g = {}^{B}T_G\,{}^{G}T_C\,{}^{C}T_g \tag{5.4}$$

$$[x_d, y_d, z_d] = [\,{}^{B}T_{g02},\ {}^{B}T_{g12},\ {}^{B}T_{g22}] \tag{5.5}$$

where $^{C}T_g$ is the transformation matrix of the detected grasp pose in the hand camera frame, $^{B}T_g$ is the transformation matrix of the detected grasp pose in the robot base frame, $^{B}T_G$ is the transformation matrix of the gripper frame in the robot base frame, and $^{G}T_C$ is the transformation matrix of the hand camera frame in the robot gripper frame, which is a constant transformation

found through the camera external calibration described in Chapter 4. In equation (5.5), ${}^{B}T_{gij}$ is the entry of ${}^{B}T_{g}$ at the *i*th row and the *j*th column.

After the grasp poses are categorized, the robot workspace limits are first used to filter out the unreachable grasp poses in each direction category. Then, the best-quality grasp in each grasp category is saved, and the other grasp poses are discarded. The robot workspace limits are the positional limits measured when the robot gripper orientation is fixed according to the grasp direction categories. Therefore, each direction category has a different workspace limit, and all are subsets of the robot's dexterous workspace. The grasp quality score generated by the grasp detection network is used as the yardstick in the grasp quality-based filtering. Figure 5.5 shows examples of the initially detected and filtered grasp poses. The pseudo-code of the whole grasp pose filtering process is presented in Table 5.3.

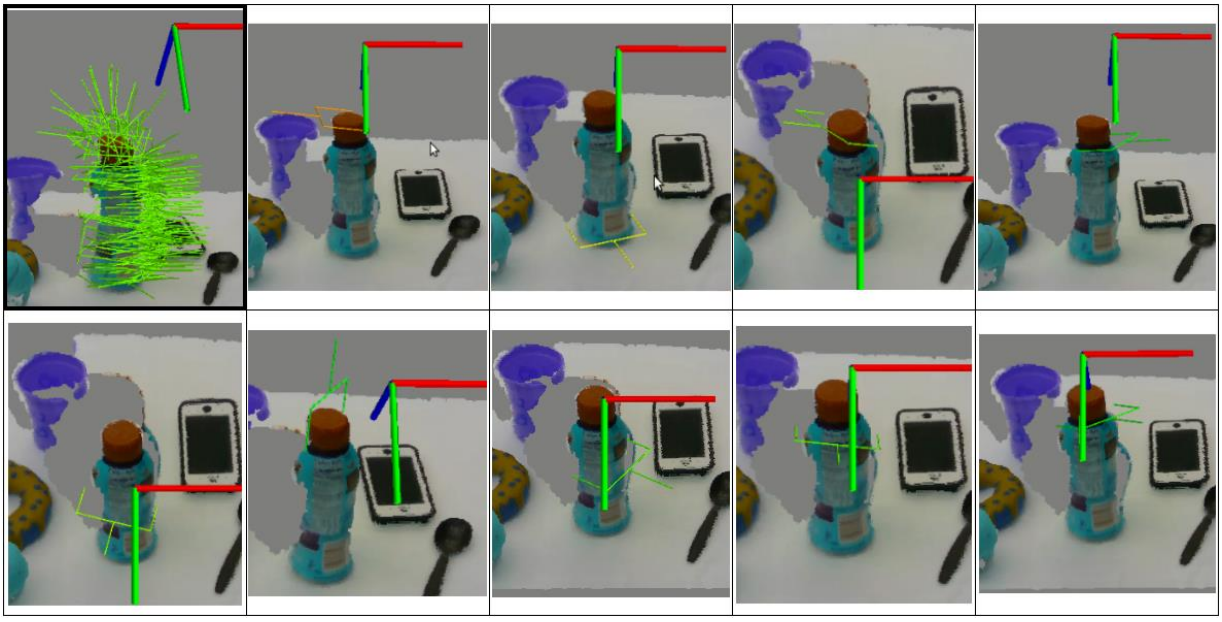


Figure 5.5 A visualization of the initially detected and filtered grasp poses. The top right image shows all detected grasp poses, and the others show the best ones in different direction categories.

Table 5.3 The pseudo code of the grasp pose filtering process.

Algorithm 5.1: Grasp pose filtering

INPUT*:* Grasp poses detected by the grasp detection network
OUTPUT: Categorized and filtered grasp poses

```
FOR each pose in the detected grasp poses:
    Transform the pose to robot base frame
    Find the main category of the pose using its direction vector's z-coordinate
    IF the main category is angled-down, side, or angled-up THEN:
        Find the sub-category of the pose using
        its direction vector's x and y-coordinates
    END IF
END FOR
FOR the poses in each grasp pose category:
    Remove the poses outside of the robot workspace
    Select the best quality pose from the poses within the robot workspace
END FOR
```

Chapter 6: Grasp Pose Refinement Module

Optimizing the grasp accuracy of the robot's grasping system is especially important for assistive robots that help people perform tasks of daily living. In this work, the grasp pose accuracy is measured by the amount of object movement caused by the robot gripper when the object is grasped. Improving grasp accuracy helps prevent problems such as spilling the content in the grasped container or losing track of the object pose in the post-grasp task. The grasp refinement module aims to find a refinement transformation matrix that can transform the initial grasp pose to a more accurate grasp pose when the initial grasp is not accurate enough. In mathematical terms, the grasp refinement process can be described in equation (6.1):

$$^{B}T_{R} = {}^{B}T_{I}\ {}^{I}T_{R} \tag{6.1}$$

where $^{X}T_{Y}$ is the transformation matrix of frame {Y} relative to the base frame {X}. The letters B, R, and I refer to the robot base frame {B}, the refined gripper frame {R}, and the initial gripper frame {I}, respectively. Thus, the terms $^{B}T_{I}$ and $^{B}T_{R}$ are the initial and the refined grasp poses, and the term $^{I}T_{R}$ is the refinement transformation matrix. Besides the form of transformation matrix, the grasp pose can also be described by a vector of six parameters describing its Cartesian position (x, y, z) and Euler angle-based orientation $(\theta_x, \theta_y, \theta_z)$. Therefore, the refinement matrix can be determined by finding the changes in the grasp pose parameters $(dx, dy, dz, d\theta_x, d\theta_y, d\theta_z)$.

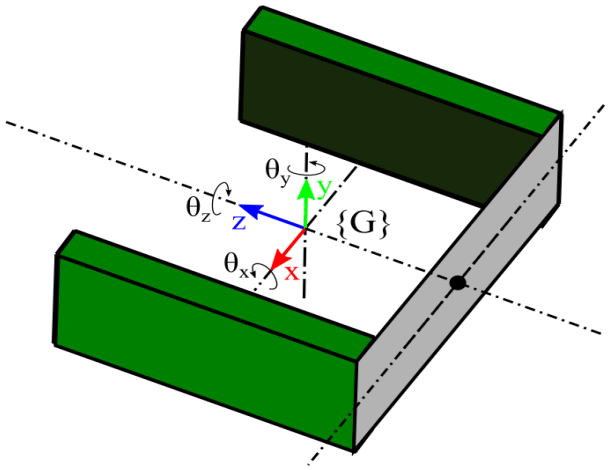


Figure 6.1 Illustration of the gripper frame assignment.

For a standard parallel-jaw gripper's coordinate frame {G}, as shown in Figure 6.1, the six pose parameters of the gripper can be classified into the grasp depth (z), the grasp direction (θ_x, θ_y), and the pose projection parameters (x, y, θ_z). The grasp depth describes how deep the gripper grabs the object in the grasp direction, and the grasp direction is defined by the direction vector of the gripper's z-axis. When the grasp direction and depth are determined, the grasp pose can be projected into an image space that is perpendicular to the grasp direction, with the grasp depth as the projection distance. The grasp pose parameters are reduced to (x, y, θ_z) in the projection image space. This pose parameter dissection helps break down the six-dimensional grasp refinement problem into a two-dimensional grasp direction refinement problem, a one-dimensional grasp depth refinement problem, and a three-dimensional pose projection refinement problem. Therefore, the grasp refinement matrix ${}^{I}T_{R}$ can be defined as the product of the grasp direction refinement matrix ${}^{I}T_{DR}$, the pose projection coarse adjustment matrix ${}^{DR}T_{PC}$, the grasp depth refinement matrix ${}^{PC}T_{DP}$, and the pose projection fine-tuning matrix ${}^{DP}T_{PF}$, as shown in equation (6.2).

$$ {}^{I}T_{R} = {}^{I}T_{PF} = {}^{I}T_{DR}\,{}^{DR}T_{PC}\,{}^{PC}T_{DP}\,{}^{DP}T_{PF} \tag{6.2} $$

In summary, the first step in the grasp refinement process is to refine the initial grasp pose's direction ($d\theta_x, d\theta_y$); the second step is to adjust the pose projection parameters ($dx, dy, d\theta_z$) based on a set of collision avoidance criteria to optimize the pose for grasp depth refinement; after this coarse pose projection adjustment, the third step is to refine the grasp depth (dz); and the last step is to finetune the pose projection parameters ($dx, dy, d\theta_z$) using the object movement-based quality measure presented in Chapter 3.

6.1 Grasp Direction Refinement

The grasp direction refinement is to find the adjustments for the gripper's pitch ($d\theta_x$) and yaw ($d\theta_y$) based on the geometry of the object's grasped part. The refinement goal is to achieve an optimal match between the gripper and the object's contact and approach surfaces, as shown in Figure 6.2. For pitch, the refinement criterion is that the grasp direction vector should be perpendicular to the *object approach surface*, which is the object surface that the gripper palm approaches. For yaw, the refinement criterion is that the gripper fingers' closing vectors should be perpendicular to the *object contact surfaces*, which are the object surface parts touched by the gripper fingers. Therefore, finding the pitch refinement angle is equivalent to finding the object approach surface's angle of inclination with respect to the gripper frame's y-axis; and finding the yaw refinement angle is equivalent to finding the object contact surfaces' angles of inclination with respect to the gripper frame's z-axis. In addition, the refinement angle of each parameter should be limited by a maximum threshold to prevent drastic grasp direction change caused by data noise. This limitation also helps to keep the grasp direction in the range set by the post-grasp task or the user's preference in the initial grasp planning stage.

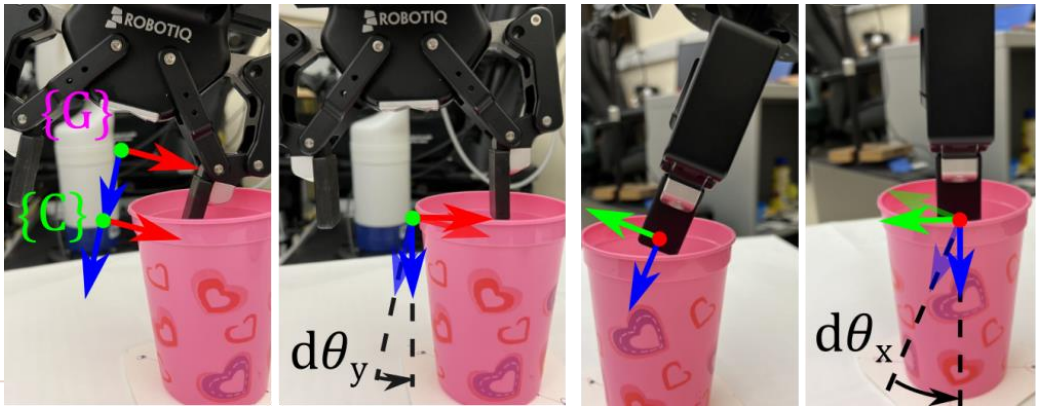


Figure 6.2 Examples of grasp pitch and yaw refinements in robot grasping. From left to right, the four pictures represent the yaw refinement (1 to 2) and the pitch refinement (3 to 4), respectively.

For computational efficiency, the regions of interest (ROIs) for the pitch and yaw refinements are defined to crop the point cloud so that the computations are constrained to the local region of the object's grasped part. In addition, to improve the accuracy of the ROI-based refinement, the refinement process is performed with multiple iterations to account for the point cloud data change in the ROIs after each grasp direction change. The grasp direction refinement can be formulated as equations (6.3) and (6.4).

$$ {}^{I}T_{DR} = {}^{I}T_{IC}\left(\prod_{i=1}^{n} {}^{YR_{i-1}}T_{YR_i}\right){}^{YR_n}T_{DR} \tag{6.3} $$

$$ {}^{YR_{i-1}}T_{YR_i} = {}^{YR_{i-1}}T_{PR_i}\,{}^{PR_i}T_{YR_i} \tag{6.4} $$

$$ when\ i = 1, YR_{i-1} \equiv IC $$

where I, IC, PR, YR, and DR represent the initial gripper frame, the initial contact center frame, the pitch-refined contact center frame, the yaw-refined contact center frame, and the direction-refined gripper frame, respectively. n is the number of iterations, and i is the iteration index.

Equation (6.4) shows the grasp direction refinement process in each iteration, which is first refining the pitch angle and then refining the yaw angle. Therefore, each iteration ends with a yaw-refined grasp pose (YR_i), and the starting pose of each iteration is the yaw-refined pose from the previous iteration (YR_{i-1}). For the first iteration (i=1), the previous yaw-refined pose is the initial grasp pose represented by its contact center frame (${}^{I}T_{IC}$= ${}^{I}T_{YR_0}$). After all iterations are finished, the contact center frame-based pose representation is transformed back to the gripper frame-based representation using ${}^{YR_n}T_{DR}$ in equation (6.3). As shown in equations (6.5) and (6.6), ${}^{YR_{i-1}}T_{PR_i}$ and ${}^{PR_i}T_{YR_i}$ are the transformation matrices of rotations about the gripper contact center frame's x-axis and y-axis, by angles $d\theta_{xi}$ and $d\theta_{yi}$, respectively.

$$ {}^{YR_{i-1}}T_{PR_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos d\theta_{xi} & -\sin d\theta_{xi} & 0 \\ 0 & \sin d\theta_{xi} & \cos d\theta_{xi} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.5} $$

$$ {}^{PR_i}T_{YR_i} = \begin{bmatrix} \cos d\theta_{yi} & 0 & \sin d\theta_{yi} & 0 \\ 0 & 1 & 0 & 0 \\ -\sin d\theta_{yi} & 0 & \cos d\theta_{yi} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.6} $$

The reason for introducing the contact center frame as the reference frame for the grasp direction refinement is that changing the grasp direction with respect to this frame does not change the grasp contact points. Figure 6.3 illustrates the difference between the results of changing the grasp direction with respect to the gripper frame and the contact center frame. This figure also shows the definition of the contact center frame, which is the gripper frame translated along its z-axis to the center of the gripper's actual contact points. The transformation matrix of the initial contact center frame in the initial gripper frame ${}^{I}T_{IC}$ is calculated using equation (6.7).

$$ {}^{I}T_{IC} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.7} $$

where d_c is the distance between the gripper frame and the contact center frame. For the Contact-GraspNet [25] detected initial grasp poses and the ROBOTIQ 2F-85 gripper [50], d_c can be set as 0.03 meters. After the formulation of the grasp direction refinement, the problem boils down to finding the pitch and yaw refinement angles $d\theta_x$ and $d\theta_y$, which are related to the object approach surface's angle of inclination with respect to the gripper frame's y-axis; and the object contact surfaces' angles of inclination with respect to the gripper frame's z-axis, respectively.

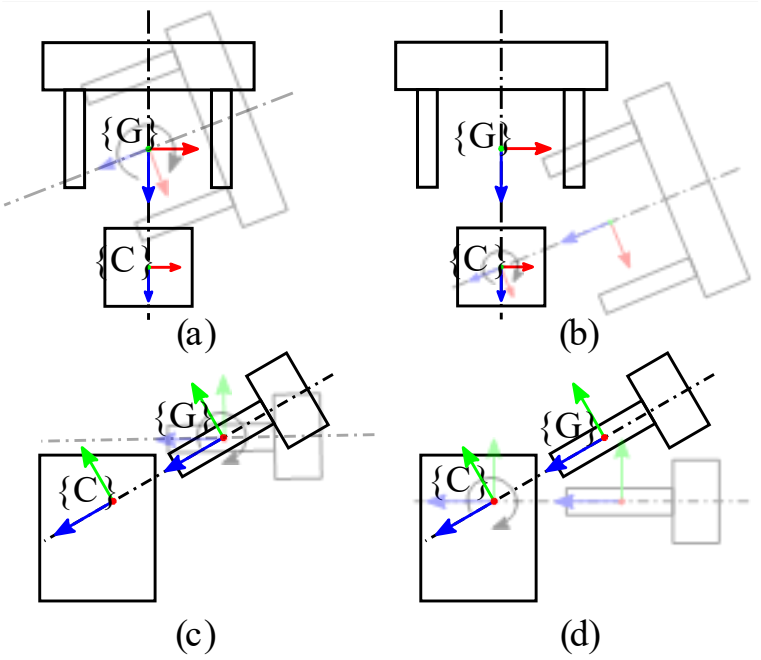


Figure 6.3 Grasp direction change with respect to the gripper frame {G} and the contact center frame {C}, respectively. (a) and (c): rotation about gripper frame; (b) and (d): rotation about contact center frame.

6.1.1 Object Surface's Angle of Inclination Estimation

The input object data to the refinement system is the combined point cloud from two different viewpoints. Because of the noise in the original point clouds and the noise introduced by the imperfect alignment between the two point clouds, the estimated normals at the surface points are also noisy, which makes the estimated angles of inclination not reliable for the grasp direction refinement. The point cloud outlier removal and point normal estimation methods from Open3D [51] are tested on the input object point cloud data, but the results are unsatisfactory. Therefore, a new method is developed to estimate the object surfaces' angles of inclination for the grasp direction refinement.

The first step in estimating the object surface's inclination angle is defining the region of interest (ROI) to crop the input point cloud. Because the input grasp pose to the grasp refinement system is assumed to be close to the object of interest, the ROIs for grasp refinement can be selected using the gripper's grasp region as a reference. As shown in Figure 6.4, the gripper's grasp region is the yellow volume. The grasp region can be described by its lower and upper limits in the gripper frame by equation (6.8).

$$V_{gr} = \begin{bmatrix} -w_{gr}/2 & w_{gr}/2 \\ -h_{gr}/2 & h_{gr}/2 \\ z_{gr1} & z_{gr2} \end{bmatrix} \tag{6.8}$$

where w_{gr} and h_{gr} are the width and height of the grasp region volume V_{gr}, which are measured along the gripper frame's x and y-axis, respectively. z_{gr1} and z_{gr2} are the lower and upper limits of the grasp region's z-coordinate. In this dissertation, the ROBOTIQ 2F-85 gripper [50] is used, and the values for the parameters w_{gr}, h_{gr}, z_{gr1}, and z_{gr2} are 0.085, 0.022, -0.032, and 0.04 in meters.

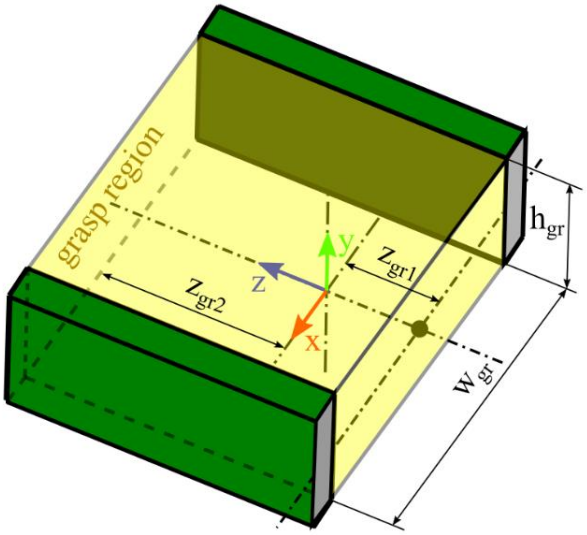


Figure 6.4 Illustration of the gripper's grasp region.

As shown in Figure 6.5, the default ROI for pitch and yaw refinements in the gripper frame can be described by equation (6.9).

$$V_{proi} = \begin{bmatrix} -w_{pyroi}/2 & w_{pyroi}/2 \\ -h_{gr}/2 & h_{gr}/2 \\ z_{gr1} & z_{pyroi} \end{bmatrix} \tag{6.9}$$

where the parameters w_{pyroi} and z_{pyroi} are empirically set to be 0.1 and 0.06 in meters. The default ROI for grasp direction refinement is an enlarged version of the grasp region volume; the enlargement is for including enough object point cloud data. In actual applications, the grasped object must be smaller than the gripper's enclosing range. Using the default ROI in the downstream calculations will waste computational power. Therefore, a refined ROI must be defined after using the default ROI to get the object point cloud data. Here we used the Open3D library [51] to get the refined ROI as the bounding box of the object point cloud in the default ROI. Through this ROI

size reduction, the amount of empty space inside the ROI is minimized, and the computational power spent on checking the empty volumes is saved.

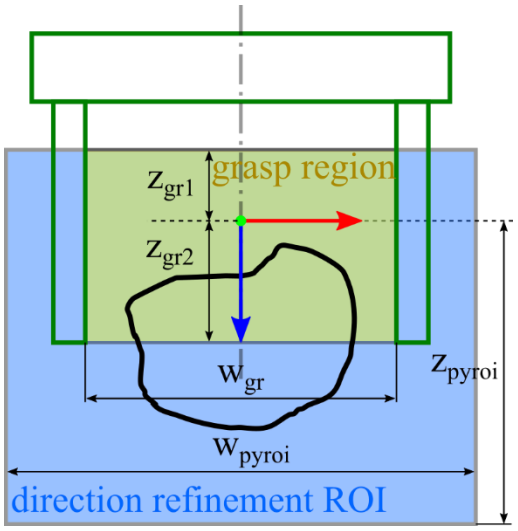


Figure 6.5 Illustration of the default grasp direction refinement ROI.

After the ROI is defined, the next step is finding the object surface's inclination angle. The object surface is first discretized, then its inclination angle is estimated by analyzing the inclination angles of the discretized surface pieces. Discretizing the object's surface simplifies finding the surface points and helps isolate the point cloud outliers. The surface points are extracted for each surface piece, and then the angle of inclination is calculated based on the surface points. After the inclination angles of all surface pieces are found, a set of criteria is used to remove outlier pieces and find the average angle of inclination of the whole surface of interest.

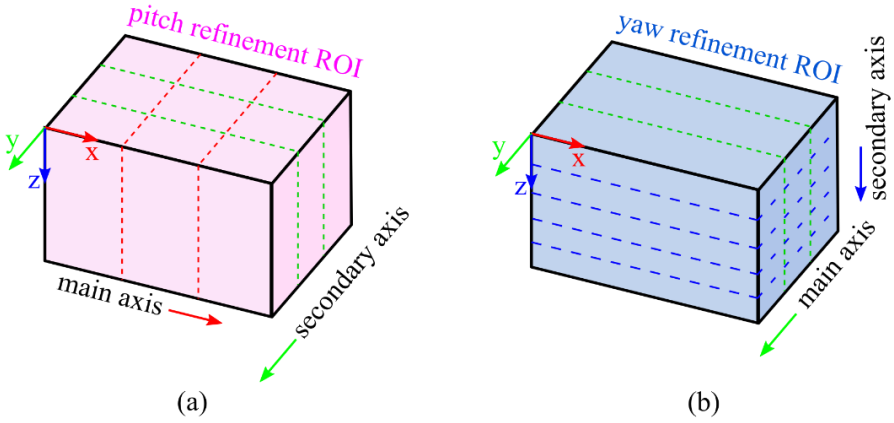


Figure 6.6 The discretized ROIs for estimating the object surfaces' angles of inclination. (a) ROI for pitch refinement, (b) ROI for yaw refinement.

In the object surface discretization, the ROI is sliced along two axes. In this work, the resultant volumes from slicing the ROI along the first (main) discretization axis are called the first-level volumes. The resultant volumes from separating the first-level volume along the second discretization axis are called the second-level volumes. The main discretization axis is in the direction of the refined parameter's rotation axis. For example, the rotation axis of pitch is the contact center frame's x-axis which is in the same direction as the x-axis of the ROI. Therefore, the main axis of discretization for the pitch refinement ROI is the x-axis. Analogically, the main axis of discretization for the yaw refinement ROI is the y-axis. Figure 6.6 shows the discretized ROIs for the pitch and yaw refinements.

The first level of discretization along the main axis separates the 3D object surface into thin slices, and in each slice, the object surface can be approximated as a 2D surface outline. The second discretization level is for finding the outline points from the cloud points in each first-level volume. As shown in Figure 6.6, because the pitch refinement depends on the object approach surface and the yaw refinement depends on the object contact surfaces, the second discretization

axes for pitch and yaw are the ROI's y and z axes, respectively. Once the ROI is discretized, the object surface points can be found as the outmost points in each second-level volume along the un-discretized axis. For pitch refinement, the point with the smallest z-coordinate in the gripper frame, in each second-level volume, is the point on the object approach surface. For yaw refinement, the point cloud's left- and right-most points in the first-level volume are found and used as the reference points of the left and right contact surfaces. Then, in each second-level volume, the points with the largest and smallest x-coordinate in the gripper frame are used to compare with the left and right reference points. If the smallest/largest point is closer to the left reference point in the x-axis direction, then the point is assigned to the left contact surface. Otherwise, it is assigned to the right contact surface.

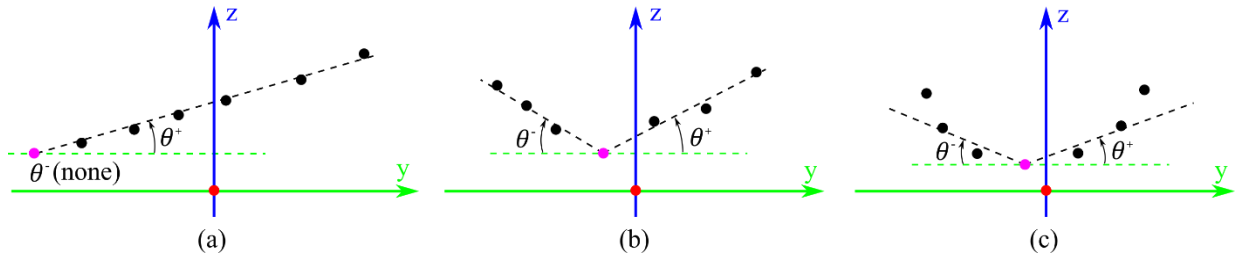


Figure 6.7 Examples of the positive and negative surface inclination angles of different approach surfaces. The black points are the surface points projected to the y-z plane along the x-axis. The pink point is the separation point for the positive and negative sides.

After the surface points are found, the surface is separated into two parts, and the angle of inclination of each part is calculated separately. For the approach surface, the surface points are separated by the closest point to the gripper palm, which is the point of the smallest z-coordinate in the gripper frame. As shown in Figure 6.7, the surface points are projected to the y-z plane along the x-axis, the pink point is the closest point to the gripper palm, and θ^- and θ^+ are the angles of inclination of the left and right surface parts. The gripper pitch needs to change in the negative

direction by the angle θ^- to align the grasp direction to the normal of the left part of the surface, and the same alignment to the right part needs a positive pitch change of angle θ^+. Similarly, for yaw refinement, the positive and negative inclination angles of each side of the contact surface are calculated based on the surface points projected to the x-z plane along the y-axis. The separation point for the contact surface is the point of the smallest/largest x-coordinate in the gripper frame for the left/right contact surface. Once the object surface is separated into two segments, the angle of inclination of each segment is obtained using a statistical method.

The first step in estimating a surface segment's inclination angle (θ^- or θ^+) is calculating the surface point vectors, starting from the separation point to every other point within the surface segment. The second step is calculating the angles of the surface point vectors with respect to the second discretization axis (y-axis for the approach surface, z-axis for the contact surfaces). The last step is estimating the angle of inclination of the surface segment using Kernel Density Estimation (KDE) [52] based clustering to find the smallest cluster center of the angle clusters. Here the statistical method is used to moderate the effect of outliers in the noisy input object point cloud. The equations for calculating the surface point vectors (V_i) and their angles (α_i) are presented in equations (6.10) and (6.11). The KDE clustering is performed using the Kernel Density function (with the Gaussian kernel) from the scikit-learn python package [53].

$$V_i = \begin{bmatrix} \Delta a_i \\ \Delta b_i \end{bmatrix} = \begin{bmatrix} a_i \\ b_i \end{bmatrix} - \begin{bmatrix} a_s \\ b_s \end{bmatrix} \tag{6.10}$$

$$\alpha_i = \arctan\left(\frac{\Delta b_i}{\Delta a_i}\right) \in [-\frac{\pi}{2}, \frac{\pi}{2}] \tag{6.11}$$

where V_i is the vector from the separation point (a_s, b_s) to the i^{th} surface point (a_i, b_i). The (a, b) coordinates are the (y, z) and (z, x) coordinates of the surface points in the gripper frame for the

approach surface and the contact surfaces, respectively. If there are not enough data points in the surface segment, the angle of inclination of the segment is set to a special value (-2 and 2 for θ^- and θ^+, respectively) which can be identified later in other calculations.

After the positive and negative inclination angles are calculated for the surface slices in each first-level volume, the next step is to estimate the whole surface's positive and negative inclination angles. The KDE method is used again to cluster the inclination angles of the surface slices in every first-level volume. Once the inclination angles of the approach surface and the contact surfaces are calculated, the pitch and yaw refinement angles can finally be calculated.

6.1.2 The Pitch and Yaw Refinement

The object approach surface points are extracted based on the initial pose, and the approach surface's positive and negative inclination angles (θ_a^-, θ_a^+) are calculated. Then, the pitch refinement angle $d\theta_{xi}$ can be found using equations (6.12) and (6.13).

$$d\theta_{xi} = \min\left(d\theta_x', \theta_{pmax}\right) \tag{6.12}$$

$$d\theta_x' = \begin{cases} 0, & \left||\theta_a^-| - \theta_a^+\right| \leq 0.175 \\ minmg(\theta_a^-, \theta_a^+), & Otherwise \end{cases} \tag{6.13}$$

Equation (6.12) aims to limit the pitch refinement by a maximum refinement angle θ_{pmax} so that the refined grasp pose does not change too much from the initial pose. In this work, we set the maximum iterations for the grasp direction refinement to be 4, and we empirically determined the maximum allowed pitch change to be 35 degrees based on the accuracy of the initial pose (When using a different grasp detection network, this number should be adjusted accordingly). Based on the maximum allowed pitch change and the number of iterations, θ_{pmax} is determined as each iteration's maximum pitch refinement angle, which is 0.1527 radians (or $\frac{35}{4}$ degrees).

Equation (6.12) presents the pitch refinement angle $d\theta_x'$ predicted from the approach surface inclination angles. The function *minmg* is the operation of selecting the smallest magnitude input variable. The number 0.175 in equation (6.13) is the angle difference threshold in radians (or 10 degrees) for judging if two angles are similar in magnitude. The angle similarity threshold of 0.175 is empirically determined based on the accuracy of the point cloud data and the angle estimation method. The conditions for the two-piece function (6.14) are derived based on the following two rules:

1. There are two cases where the pitch refinement should be zero. One is when the two inclination angles of the approach surface (θ_a^-, θ_a^+) have similar magnitudes, which means the positive and negative parts of the surface are approximately symmetrical about the gripper's z-axis when the surface is projected onto the y-z plane. An example is when grasping a sphere and the gripper's z-axis goes through the center of the sphere, in which case the gripper pitch does not need to be changed. Case two is when no surface data is available (or the available data is too noisy). In this case, the two inclination angles are set to special values -2 and 2; since no reliable information can be used to change the pitch, it is better to keep the pitch unchanged.
2. There are also two cases where the pitch refinement should be set as the smaller-magnitude inclination angle of the approach surface. Case one is when θ_a^- or θ_a^+ has a magnitude of 2, which means one side of the surface data is absent or noisy, so the pitch refinement should be based on the other good side. Case two is when both θ_a^- and θ_a^+ are in the normal magnitude range, but their magnitudes are not close, so the pitch should change towards aligning the gripper to the closest side of the surface, if the change does not exceed.θ_{pmax}.

After the pitch is refined, the two contact surfaces' positive and negative inclination angles (left contact: θ_{lc}^-, θ_{lc}^+, right contact: θ_{rc}^-, θ_{rc}^+) are calculated based on the pitch-refined pose. The

overall positive and negative inclination angles are calculated with equation (6.14) based on the four inclination angles of the two contact surfaces. The purpose of using the overall inclination angles for yaw refinement is to achieve the goal of fitting both contact surfaces of the gripper fingers to the two sides of the object contact surfaces in the most stable condition (which is when the gripper contact surfaces' normals are aligned with the object contact surfaces normals).

$$\theta_c^s = \begin{cases} 0, & \theta_{lc}^s = -2 \; or \; \theta_{lc}^s = 2 \; or \; \theta_{rc}^s = -2 \; or \; \theta_{rc}^s = 2 \\ minmg(\theta_{lc}^s, \theta_{rc}^s), & Otherwise \end{cases} \tag{6.14}$$

The letter *s* in equations (6.14) and (6.15) is the positive or negative sign placeholder. For the positive side of the contact surfaces, *s* is +; otherwise, *s* is –. In equation (6.14), θ_c^s is the overall inclination angle of the positive or negative side of the contact surface. The first case of the equation means the overall inclination angle is set to zero if the available data is not enough to estimate both inclination angles of the left and right contact surfaces. The second case of the equation means that when both the left and right inclination angles are obtained, the overall inclination angle is set as the inclination angle of the smaller magnitude. Using equation (6.14), the positive and negative overall inclination angles (θ_c^+, θ_c^-) of the contact surfaces are calculated. Then the initial estimation of the positive and negative yaw refinement angles can be calculated using equation (6.15).

$$d\theta_y^s = \begin{cases} minmg\ (\theta_c^s, \theta_{ymax1}), & \left||\theta_{lc}^s| - |\theta_{rc}^s|\right| < 0.175 \\ minmg\ (\theta_c^s, \theta_{ymax2}), & Otherwise \end{cases} \tag{6.15}$$

where θ_{lc}^s and θ_{rc}^s represent the positive and negative inclination angles of the two contact surfaces (left contact: θ_{lc}^-, θ_{lc}^+, right contact: θ_{rc}^-, θ_{rc}^+). The first case of equation (6.15) is that the inclination angles of the left and right contact surfaces are similar, so the confidence level of the calculated

overall inclination angle is high. Thus, the initial yaw refinement angle is set to the overall inclination angle (θ_c^S) with a loose limitation (limited by a large maximum yaw refinement threshold θ_{ymax1}). The second case of the equation is that the confidence level on the calculated overall inclination angle is lower than the first case. Therefore, the limitation on the yaw refinement angle is stricter (limited by a small maximum yaw refinement threshold θ_{ymax2}). In this work, we set θ_{ymax1} and θ_{ymax2} to 0.35 and 0.17 radians, respectively. The selection of these threshold values is based on the accuracy of the initial grasp poses; if using a different grasp detection network, these values should be adjusted accordingly.

Once the initial positive and negative yaw refinement angles ($d\theta_y^-$, $d\theta_y^+$) are obtained, the final yaw refinement angle can be found using equation (6.16).

$$d\theta_{yi} = \begin{cases} maxmg(d\theta_y^-, d\theta_y^+), & d\theta_y^- = 0 \; or \; d\theta_y^+ = 0 \\ 0.5(d\theta_y^- + d\theta_y^+), & \left| d\theta_y^+ - \left| d\theta_y^- \right| \right| < 0.175 \\ minmg(d\theta_y^-, d\theta_y^+), & Otherwise \end{cases} \tag{6.16}$$

where *maxmg* is the operation of selecting the largest magnitude input value. The first case of equation (6.16) is when one of the positive or negative refinement angles is zero; then the final refinement angle is equal to the none-zero one; if both refinement angles are zero, then the final refinement angle is equal to zero as well. The second case is when the positive and negative refinement angles' magnitudes are close and nonzero, which means the contact surface is approximately symmetrical about the surface normal at the surface separation point. In this case, the yaw refinement equals the average of the positive and negative refinement angles. Averaging the angles leads the gripper grasp contact surface normal to be aligned with the average surface normal at the surface separation point. The third case is when the positive and negative refinement angles are nonzero, and their magnitudes are not close. The final yaw refinement angle is set as

the angle of a smaller magnitude. After the yaw refinement angle is obtained, the yaw refinement matrix can be calculated. The yaw-refined grasp pose can be obtained by multiplying the pitch-refined grasp pose matrix by the yaw refinement matrix. Once $d\theta_{xi}$, $d\theta_{yi}$, the pitch and yaw refinement matrices are obtained, one iteration of the grasp direction refinement is completed.

6.2 Pose Projection Parameters Coarse Adjustment

After the grasp direction is refined, the next step is to coarsely adjust the pose projection parameters to clear the path of the gripper in the grasp direction so that the chance of the gripper reaching the desired grasp depth is maximized. The examples shown in Figure 6.8 illustrate how the initial grasp pose can affect the result of grasp depth refinement. In the stage of coarse adjustment, the pose parameters (x, y, θ_z) are modified based on the following collision avoidance criteria:

1. The gripper roll (θ_z) should be modified first to make the gripper and object contact surfaces as parallel as possible.
2. The parameter x should be changed in the x-axis direction so that the gripper center is at the center of the object point cloud inside the gripper's grasp region bounding box. Adjusting parameters x and θ_z will maximize the empty space in the path of the gripper fingers when the gripper moves along the grasp direction to grab the object.
3. The parameter y should be changed to move away from the objects in the surroundings that block the grasping path.

The pose projection parameters' coarse adjustment matrix ${}^{DR}T_{PC}$ is calculated using equation (6.17).

$$ {}^{DR}T_{PC} = T_{roll}T_xT_y \tag{6.17}$$

where T_{roll}, T_x, and T_y are the transformation matrices of the grippe roll change ($d\theta_z$), x-coordinate change (dx), and y-coordinate change (dy), respectively. Equations (6.18) to (6.20) present the explicit form of T_{roll}, T_x, and T_y.

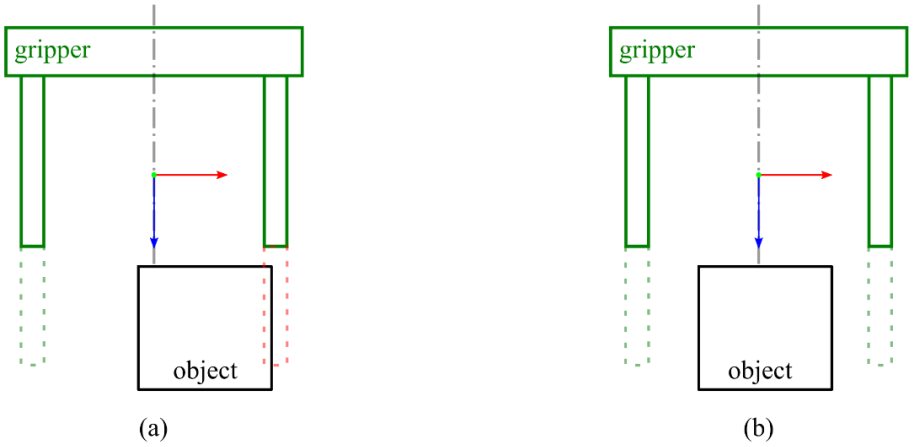


Figure 6.8 Illustration of how the initial pose affects the grasp depth refinement. (a) The initial pose position is not adjusted for collision avoidance. In this case, the right-side gripper finger will collide with the object if the gripper needs to reach deep enough to grasp the object. (b) The grasp pose is adjusted to a good position where the gripper can reach down to the desired grasp depth.

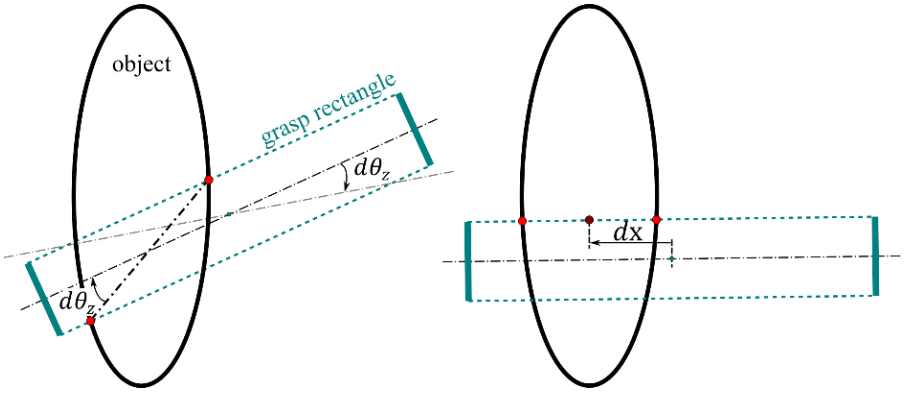


Figure 6.9 Illustration of the coarse adjustments of the gripper's roll and x-position. The red points are the primary contact points. $d\theta_z$ and dx are the roll and x-position adjustments, respectively.

The first step of finding the gripper roll adjustment is to crop the object point cloud using the gripper grasp region ROI and extract two primary contact points from the object points inside the ROI. The object's primary contact points are the two closest object surface points to each of the gripper fingers' contact surfaces. As shown in Figure 6.9, the roll change $d\theta_z$ is the angle between the gripper's x-axis and the vector from the right-side primary contact point to the left-side primary contact point. The grasp roll adjustment matrix is calculated by equation (6.18). Because the roll adjustment will change the object point cloud inside the gripper's grasp region, the roll adjustment must be performed with multiple iterations. The iteration number is set to 3 in this work.

$$T_{roll} = \begin{bmatrix} \cos d\theta_z & -\sin d\theta_z & 0 & 0 \\ \sin d\theta_z & \cos d\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.18}$$

After the grasp roll adjustment is completed, the same procedure is used to extract the current grasp pose's primary contact points. After obtaining the new primary contact points, the approximate center of the object inside the gripper's grasp region can be calculated as the average of the two primary contact points. As shown in Figure 6.9, the x-coordinate adjustment dx is then calculated as the difference between the approximate object center point's x-coordinate and the grasp pose's current x-coordinate. The grasp pose's x-coordinate adjustment matrix can be calculated by equation (6.19).

$$T_x = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.19}$$

The robot gripper and the robot forearm link's collision bounding boxes are used for collision checking to find the y-coordinate adjustment dy. No adjustment is needed (dy=0) if there is no object point inside the collision boxes. Otherwise, the y-coordinate is changed to move away from the object points inside the robot collision boxes. The y-coordinate adjustment matrix can be calculated by equation (6.20).

$$T_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.20}$$

6.3 Grasp Depth Refinement

The goal of grasp depth refinement is to avoid collision and slipping in robot grasping. As shown in Figure 6.10, two types of collision could occur when the grasp is too deep. One is that the gripper palm could collide with the grasped object (a). Another is that the gripper fingertips could collide with the object behind the grasped object (b). If the grasp is too shallow (c), the object could slip out of the gripper.

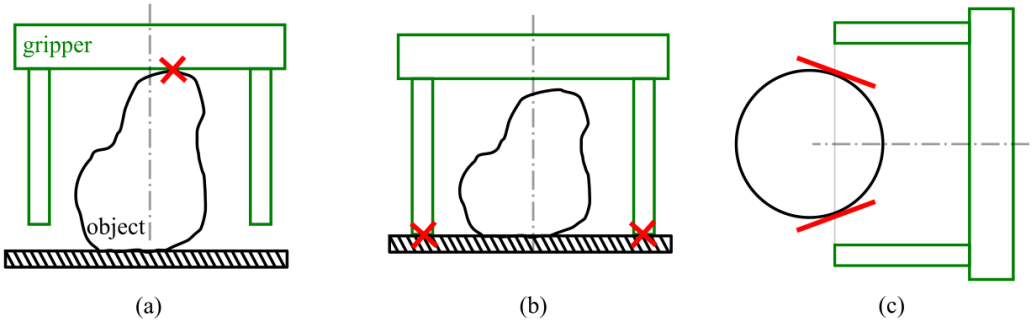


Figure 6.10 Examples of grasps with inappropriate grasp depth. (a) and (b): collision occurred because the grasp is too deep; (c): the object will slip out of the gripper because the grasp is too shallow.

The grasp depth refinement must be less than a maximum threshold dz_{max} to avoid collisions. Let dz_{max1} be the maximum allowed depth increase to avoid collisions at the gripper palm, and dz_{max2} be the maximum allowed depth increase to avoid collision around the gripper fingers. Then the grasp depth refinement's maximum threshold dz_{max}, as equation (6.21), is selected as the smaller one of the two.

$$dz_{max} = min\ (dz_{max1}, dz_{max2}) \tag{6.21}$$

Besides collision avoidance, the grasp depth also must be changed to ensure the object can be stably grasped. We use a minimum threshold dz_{min}, defined by equation (6.22), to prevent empty grasps and a use desired grasp depth d_{gd} to find the best refinement value $dz_{desired}$, as shown in equation (6.23).

$$dz_{min} = oz_{min} - g_{fz} \tag{6.22}$$

$$dz_{desired} = dz_{min} + d_{gd} \tag{6.23}$$

In equation (6.22), oz_{min} is the z-coordinate of the closest object point to the gripper palm, in the gripper frame. g_{fz} is the z-coordinate of the gripper fingertips in the gripper frame, which is a constant. d_{gd} is empirically picked, based on the size of the target objects, to define how much depth in the object should be grasped in the z-direction of the gripper frame. d_{gd} is set to 0.03 meters in this work. After calculating the desired grasp depth, the final grasp depth refinement can be determined using equation (6.24).

$$dz = min(dz_{desired}, dz_{max}) \tag{6.24}$$

The first step in finding the parameters dz_{max1}, dz_{max2}, and oz_{min} from the object point cloud data is to define the ROI to extract the object points. As shown in Figure 6.11, the ROI for grasp depth refinement is composed of three parts which are defined by the coordinates in equations (6.25) to (6.27).

$$V_{droi1} = \begin{bmatrix} -w_{gr}/2 & w_{gr}/2 \\ -h_{gr}/2 & h_{gr}/2 \\ -g_{pz} & 2g_{fl} \end{bmatrix} \tag{6.25}$$

$$V_{droi2} = \begin{bmatrix} w_{gr}/2 & w_{gr}/2 + g_{fw} \\ -h_{gr}/2 & h_{gr}/2 \\ -g_{pz} & oz_{min} + g_{fl} \end{bmatrix} \tag{6.26}$$

$$V_{droi3} = \begin{bmatrix} -w_{gr}/2 - g_{fw} & -w_{gr}/2 \\ -h_{gr}/2 & h_{gr}/2 \\ -g_{pz} & oz_{min} + g_{fl} \end{bmatrix} \tag{6.27}$$

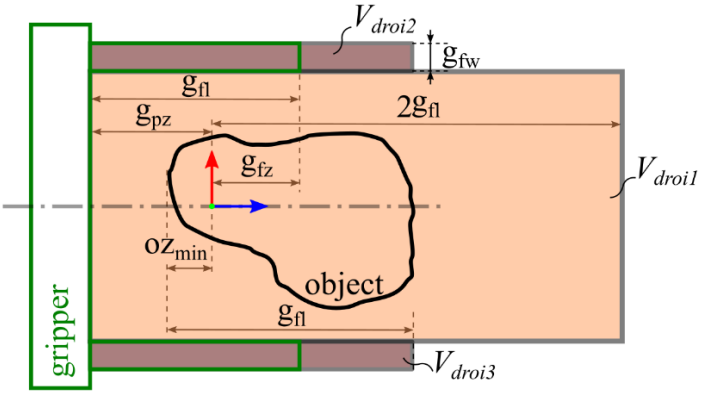


Figure 6.11 Illustration of the grasp depth refinement ROI.

V_{droi1} (ROI-1), V_{droi2} (ROI-2) and V_{droi3} (ROI-3) are the three ROIs for depth refinement. w_{gr} and h_{gr} are the grasp region's width and height as defined in Figure 6.4 and equation (6.8).

As shown in Figure 6.11, g_{fw} and g_{fl} are the width and length of the gripper finger, respectively. g_{pz} is the distance between the gripper palm and the gripper frame origin along the gripper frame's z-axis. oz_{min} is the smallest z-coordinate, measured with respect to the gripper frame, of the object points inside ROI-1.

After the grasp depth ROI and oz_{min} are evaluated, dz_{max1} and dz_{max2} can be determined using equations (6.28) and (6.29). oz_{min23} is the smallest z-coordinate, measured with respect to the gripper frame, of the object points inside ROI-2 and ROI-3. Once dz is determined, the depth refinement matrix can be computed by equation (6.30)

$$dz_{max1} = oz_{min} + g_{pz} \tag{6.28}$$

$$dz_{max2} = oz_{min23} - g_{fz} \tag{6.29}$$

$$^{PC}T_{DP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.30}$$

6.4 Pose Projection Parameters Finetuning

After the grasp direction and depth are refined, the object grasp region is determined, and the projected grasp pose parameters can be refined based on the geometry of the object's grasped part. The flowchart of the projected grasp pose refinement is presented in Figure 6.12. The projected grasp pose refinement process starts with extracting the object silhouette in the grasp direction of the initial grasp pose. With the object silhouette, the grasp pose's grasp quality features and score can be calculated; and the grasp pose can be refined based on the grasp quality features. In summary, the pose projection parameters' refinement is an iterative process containing two sub-processes: object silhouette extraction and grasp quality measure-based refinement. The

termination points of the iterative refinement process are the maximum iteration number and the desired grasp quality score threshold.

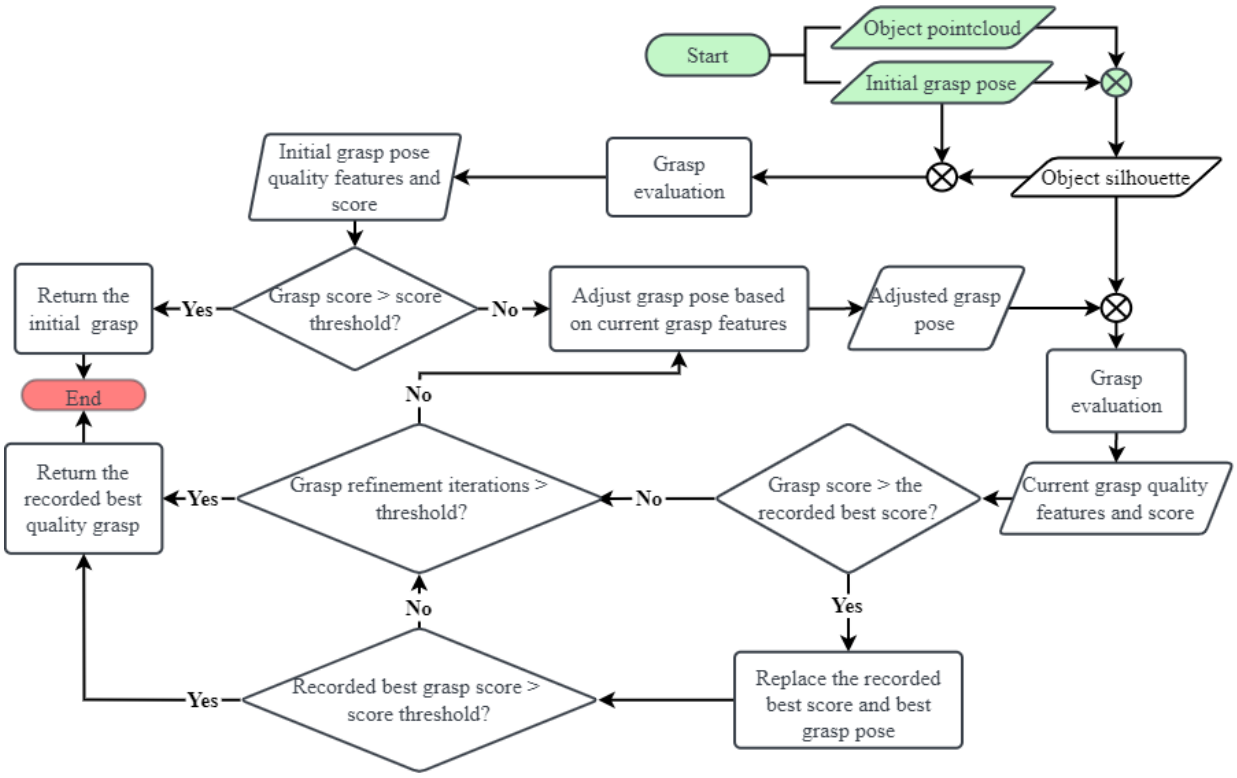


Figure 6.12 The flowchart of the grasp pose projection parameters refinement process.

6.4.1 Object Silhouette Extraction

Two methods are used in this work to extract the object silhouette for grasping. One method uses the robot vision system's object recognition and segmentation network (Chapter 4) to get the object's segmentation mask as the silhouette for grasping. This method is used when the object is small and recognizable by the vision system. For objects of relatively large size (the depth camera can differentiate the object and its background), another object point cloud-based method is used. As the first method is presented in Chapter 4, this section only presents the second object point cloud-based silhouette extraction method.

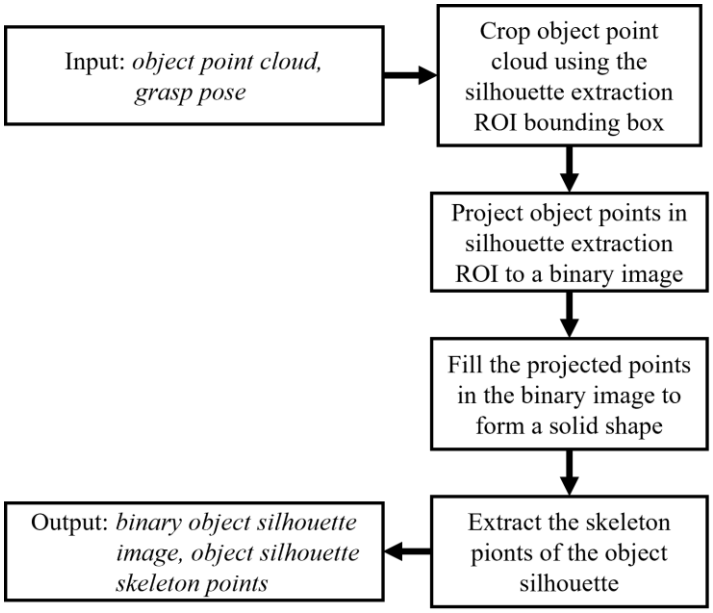


Figure 6.13 The block diagram of the object point cloud-based silhouette extraction process.

As shown in the block diagram (Figure 6.13), the first step in object silhouette extraction is to crop the object point cloud using the ROI bounding box. Like the pitch and yaw refinement ROIs, the silhouette extraction ROI can be defined based on the gripper grasp region using equation (6.31).

$$V_{sroi} = \begin{bmatrix} -w_{sroi}/2 & w_{sroi}/2 \\ -h_{sroi}/2 & h_{sroi}/2 \\ -g_{pz} & g_{fz} \end{bmatrix} \tag{6.31}$$

where w_{sroi} and h_{sroi} are the width and height of the ROI measured along its x and y axes, respectively. g_{pz} and g_{fz} are the distances of the gripper palm and fingertip to the gripper frame's origin in the z-axis direction.

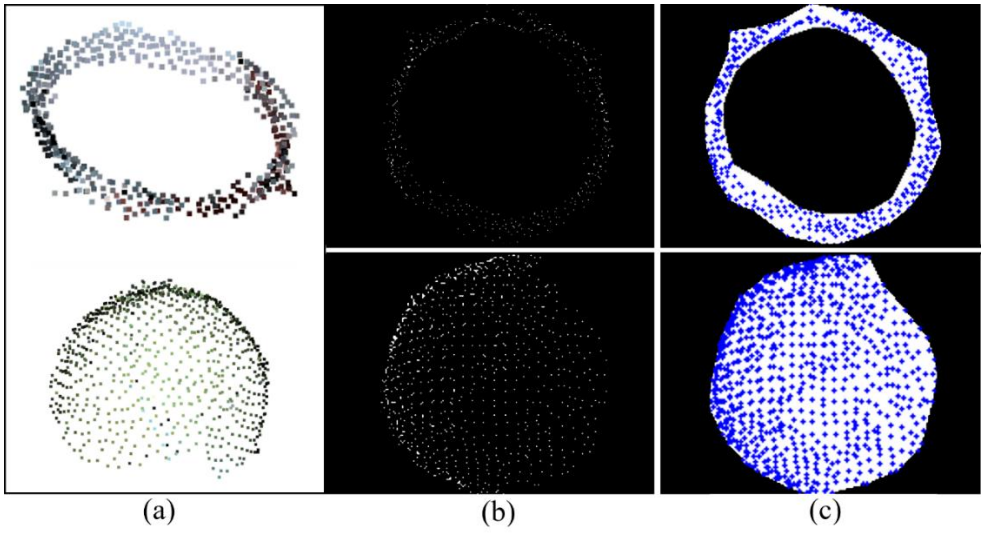


Figure 6.14 Examples of object silhouette extraction. (a): the cropped object point cloud; (b): the object points' binary projection image; (c): the extracted object silhouette. The blue points are the object points projection.

Once the silhouette extraction ROI is computed, the object points inside the ROI are projected to an image with a depth of 0.5 meters using the pinhole camera model-based 3D world coordinates to 2D image pixels mapping [54]. After the binary image of the object's projected points is acquired, the algorithm developed to fill the point image to form a solid silhouette image is presented in Table 6.1. A KD-tree [55] of the points is constructed to improve the efficiency of finding the neighboring points within a radius. The KD-tree and the convex hull-related computations are implemented using the SciPy library [49]. Figure 6.14 shows the example results of the object silhouette extraction on two objects. To reduce the search space for the grasp quality measure-based refinement, we compute the object silhouette's skeleton points using a skeletonization method [56] implemented in the scikit-image library [57]. A skeletonization test result of an object of complex shape is shown in Figure 6.15. Similarly, Vohra et al. [21] used object skeletonization to reduce the grasp pose search space and proved the effectiveness of this approach.

Table 6.1 Pseudo code for converting the object points projection image to the object silhouette image.

Algorithm 6.1: Fill object points projection image to form object silhouette image

INPUT: A binary image of object projection points
OUTPUT: A binary image of a solid shape defined by the points in the input image

```
1:  FOR each point in the input image's points:
2:      Find neighbors of the point within radius r.
3:      IF the number of neighbors > 5 THEN:
4:          Calculate the convex hull of the neighbors := neighbor_hull
5:          FOR each neighbor in the bounding box of the neighbor_hull:
6:              IF the pixel value of the neighbor is 0
                AND the neighbor is in the neighbor_hull THEN:
7:                  Set the neighbor's pixel value to 1
8:              END IF
9:      ELSE:
10:         The point is an outlier, set the pixel value of the point to 0
11:     END IF
12: END FOR
```

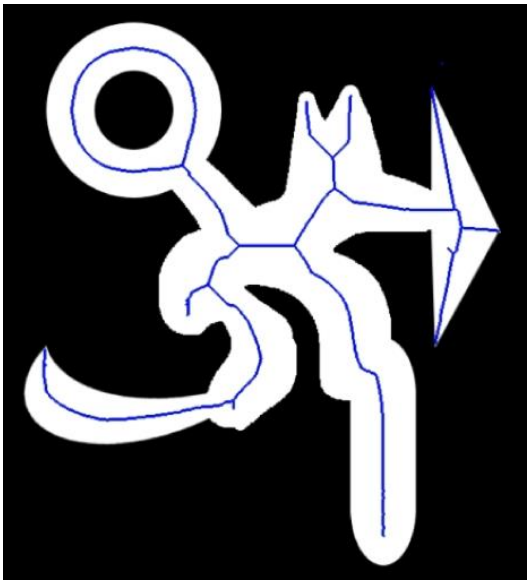


Figure 6.15 The skeletonization result of a silhouette of an imaginary object of complex shape. The blue points are the skeleton points.

6.4.2 Grasp Quality Measure-Based Refinement

As the flowchart in Figure 6.12 presents, the grasp quality measure-based refinement is a process that repeats the steps of evaluating the grasp pose, checking the grasp quality score, and modifying the grasp pose projection parameters (x, y, θ_z). The grasp pose evaluation uses the grasp quality measure presented in Chapter 3. A normalized grasp quality score and four object movement-based quality features are calculated through grasp evaluation. If the grasp quality score is less than the score threshold (0.95), the grasp pose parameters are adjusted based on the object translation and rotation features. The pseudo-code of the grasp quality measure-based refinement is presented in Table 6.2. Three example results of the grasp projection parameters refinement are shown in Figure 6.16. The grasp quality measure-based refinement depends on the accuracy of the object silhouette extracted from the object partial point cloud. Because the extracted object silhouette can be noisy, the gripper's x, y, and roll ($d\theta_z$) should be tuned again using the object point cloud after the quality measure-based refinement. The final tuning of the grasp pose's x, y, and $d\theta_z$ share the same procedures as the coarse adjustment process presented in Section 6.2. This additional point cloud-based pose tuning helps prevent any false refinement caused by noisy object silhouette and serves as a safety check before the robot performs the grasp.

Table 6.2 The pseudo code of the grasp quality measure-based refinement.

Algorithm 6.2: Grasp quality measure-based refinement

INPUT: An initial grasp pose, the object silhouette image, the object silhouette skeleton points
OUTPUT: The resultant grasp pose from refining the initial grasp pose's parameters (x, y, θ_z)

```
Sort the silhouette skeleton points by their distance to the initial grasp point ← sorted skeleton points
iteration := 0
best pose := initial pose
best score := 0
FOR point in sorted skeleton points:
    iteration + 1 := iteration
    Find the best grasp near point := current pose, current score
    IF current score > best score THEN:
        best pose := current pose
        Best score := current score
    END IF
    IF current score > 0.95 THEN:
        RETURN best pose
    END IF
    IF iteration > 100:
        Warn the user about the possibility of low-quality grasp pose
        RETURN best pose
    END IF
END FOR
```

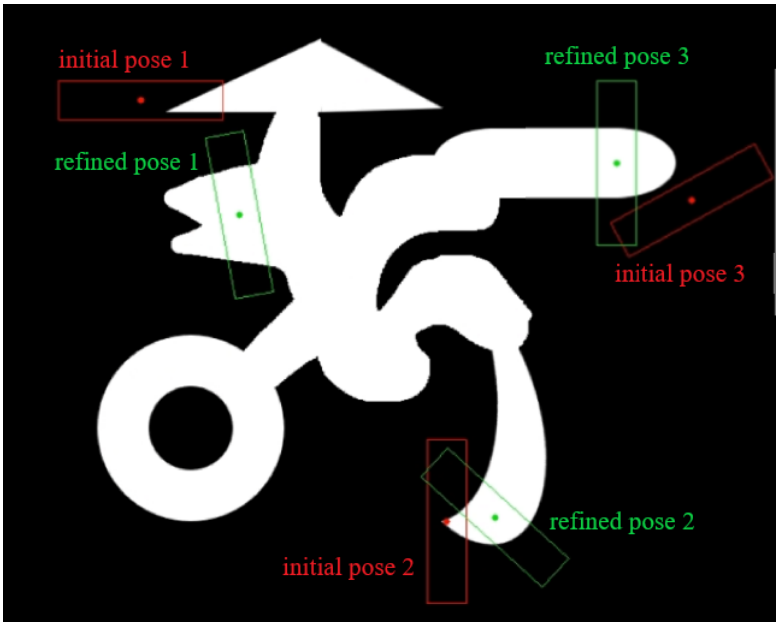


Figure 6.16 Example results of the grasp pose projection parameters refinement. The white region is the silhouette of an imaginary object of complex shape. The initial grasp poses are in red, and the refined grasp poses are in green.

Chapter 7: Human-in-the-Loop (HitL) Robot Grasping System

7.1 System Overview

After the key modules were developed, a graphical user interface and an assisted velocity control method were created to form the human-in-the-loop (HitL) robot grasping system. The system has a high autonomy grasping mode based on autonomous grasp detection and refinement and a low autonomy grasping mode based on assisted velocity control. The high autonomy mode is the default and can be switched to the low autonomy mode when the user knows the object is hard to grasp using the high autonomy mode. In terms of the objects, the grasping system can be used to grasp both known and unknown objects. Pre-defined good-quality grasp poses can be saved with the objects' point cloud models for known objects. Therefore, when the vision module retrieves a known object's 6D pose, the saved grasp poses can then be used to grasp it. In this case, the grasp detection module is unnecessary, but the grasp refinement module can still be used to ensure no collision in performing the saved grasp poses. Because the known object grasping scenario is a trivial problem in most cases, the HitL robot grasping system is developed mainly to tackle the unknown object grasping problem.

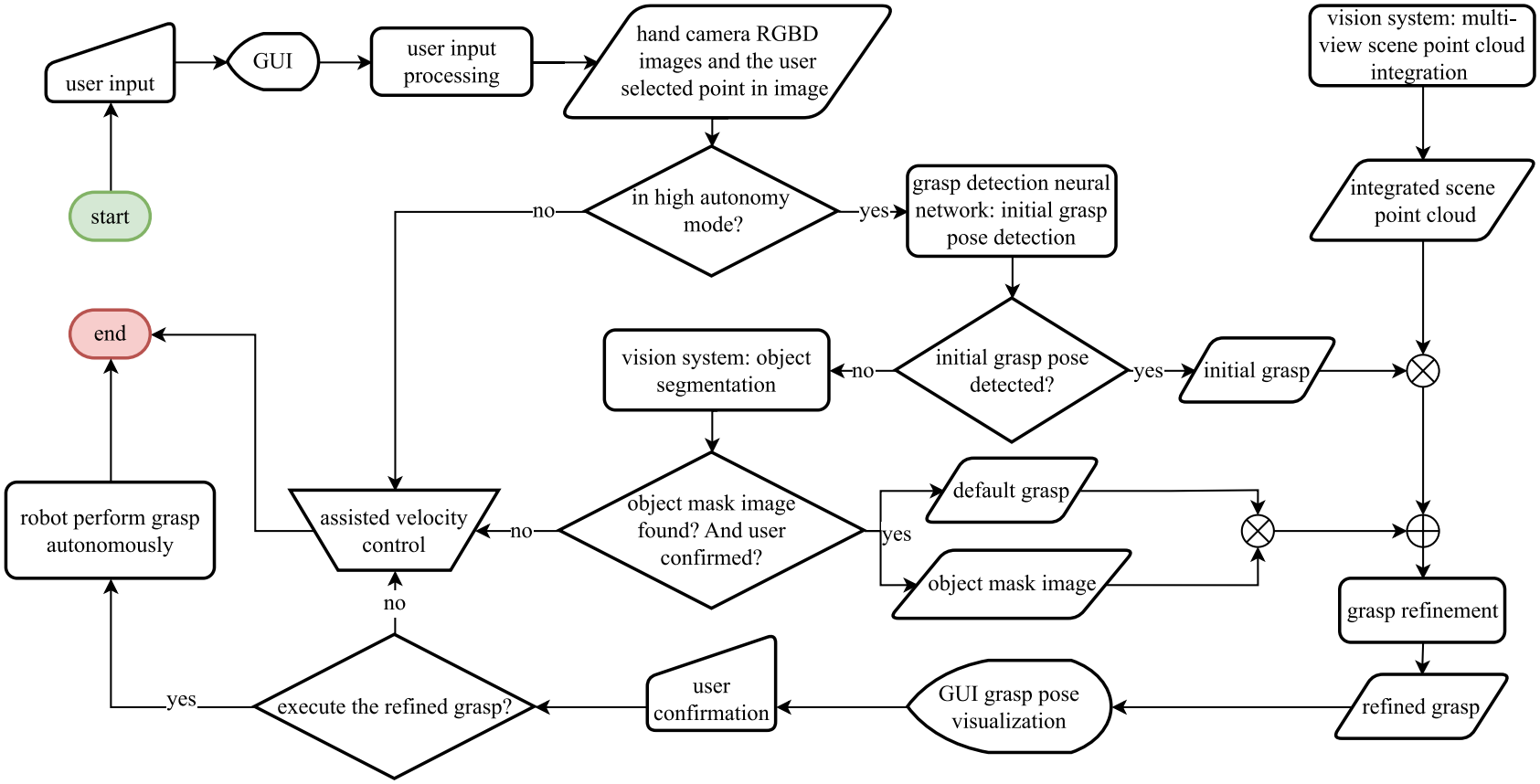


Figure 7.1 Flowchart of the human-in-the-loop robot grasping system.

As shown in Figure 7.1, the workflow of the grasping system starts with the user selecting the object to grasp from a graphical user interface (GUI). Then the user-selected point on the screen is processed to find the corresponding point in the image of the scene, which can be used to locate the object of the user's interest. After obtaining the user-selected object point, the robot gripper will move close to the user-selected object if the system is in low autonomy mode. The user can use the assisted velocity control to grasp the object after the gripper is moved close to the object. The assisted velocity control uses a virtual sphere to help adjust the gripper's direction and uses automatic mode switching to reduce the required user input (Section 7.2). If the system is in high autonomy mode, the RGBD scene images will be used by the grasp detection neural network to generate the initial grasp pose. Based on if the neural network can detect any grasp pose to grasp the user-selected object, the workflow branches to path-a (grasp pose detection succussed) and path-b (grasp pose detection failed).

Following path-a, the grasp refinement system uses the object geometry information from the multi-view scene point cloud integrated by the vision module to finetune the initial grasp pose detected by the grasp detection module. After the grasp pose is refined, it is visualized by the GUI, and the user can judge if the robot can perform the grasp pose. User supervision is the best way to guarantee safety in operating the robot with high autonomy. If the user finds the refined grasp unsatisfactory, the assisted velocity control mode can be used to grasp the object.

Following path-b, the vision system performs object segmentation using the RGB image of the scene. If the object segmentation is successful, the workflow branches to path-b1; otherwise, the workflow goes to path-b2. In path-b1, the object segmentation mask image is used as the object silhouette image. The grasp refinement system takes the integrated scene point cloud and the object silhouette image to finetune a universal default grasp pose. After the refined grasp pose is obtained,

the rest of the workflow is the same as in path-a. In path-b2, since both the grasp detection and the object segmentation have failed, the only option left is the assisted velocity control.

The system's GUI is currently developed as a desktop application with the keyboard and mouse as input devices. Any input device with at least 3-DoF inputs can control the robot grasping system. The default display of the interface is the live scene image. In the high autonomy grasping mode, when the vision module recognizes the user-selected object, the object segmentation mask image will be shown to the user as in Figure 7.2. After the grasp pose detection and refinement, the initial and refined grasp poses will be presented to the user as in Figure 7.3. After seeing the visual feedback, the user can judge if the system is working properly and can interrupt the workflow if any of the intermediate processes yield unsatisfactory results. In the low autonomy grasping mode, the user interface shows the live scene image and the textual note of the current stage of the assisted velocity control mode.

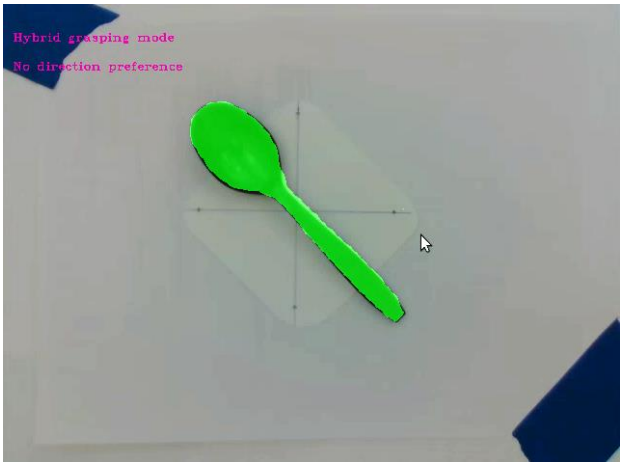


Figure 7.2 An example of the GUI display of the object segmentation result.

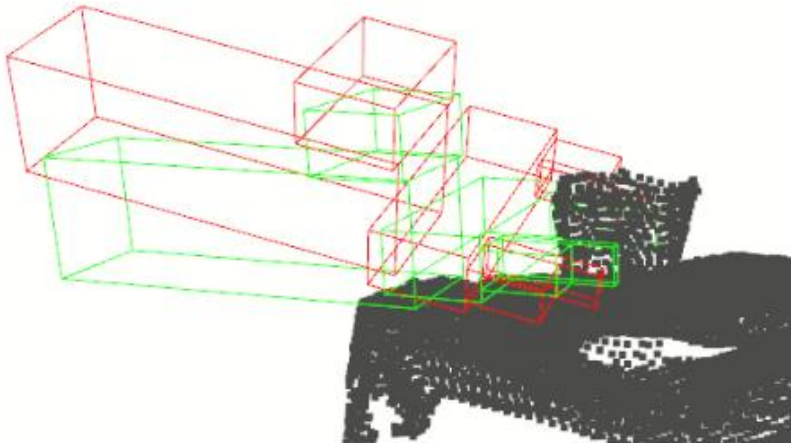


Figure 7.3 An example of the grasp pose refinement result displayed in the GUI. The robot forearm and gripper's collision boxes are drawn in red for the initial grasp pose and green for the refined grasp pose.

The Robot Operating System (ROS) [58] was used as the base framework to integrate all the components of the robot grasping system. The robot grasping system was programmed as five ROS nodes, the robot vision node, the grasp detection node, the grasp refinement node, the robot motion control node, and the main workflow manager node. The first three nodes contain the three robot grasping system modules of the corresponding names. The robot motion control node was programmed to take the robot control commands to move the physical robot. The main workflow manager node contains the user interface and workflow management modules.

7.2 Assisted Velocity Control

The high autonomy mode of the grasping system is not always reliable in complex real-life environments. Hence, a user-friendly manual control system must be added to the HitL robot grasping system to allow the user to operate the robot when the autonomous grasp planning fails. An assisted velocity control method was developed to enable the user to teleoperate the robot to grasp objects in 6D grasp pose space using three degrees of freedom input. This method uses camera depth data, virtual fixture, mixed coordinate systems, and automatic mode switching to

allow the user to use three degrees of input to control the six-dimensional grasp pose plus one more dimension for opening or closing the gripper. For comparison, the traditional teleoperation method requires at least three degrees of input to control the robot pose in a 3D position or orientation control mode, plus two more degrees of input to switch modes and open/close the gripper.

The flowchart of the assisted velocity control system is shown in Figure 7.4. The system has four different modes (pre-grasp mode, grasp direction control mode, grasp finetuning mode, and final stage mode), and three input channels are used for different control purposes in each mode. In the initial stage (pre-grasp mode), the system waits for the user to select a point on the screen to indicate the object to be grasped. The required user inputs include two degrees of freedom for moving to the point of interest in the GUI display and one input for confirmation (for the touch screen, the user can directly touch to select the object of interest). After the object point is selected on display, the actual object point in the robot base frame can be calculated by de-projecting the 2D pixel coordinates in the image to the 3D real-world Cartesian coordinates. Then based on the object point, a pre-grasp pose can be defined to move the robot gripper close to the object with a default distance and a default grasp direction. In this work, the distance is set to 0.1 meters, and the grasp direction is set to top-down or side depending on where the selected object point is. (If other object points are above the selected point, use side grasp as default; otherwise, use top-down grasp.)

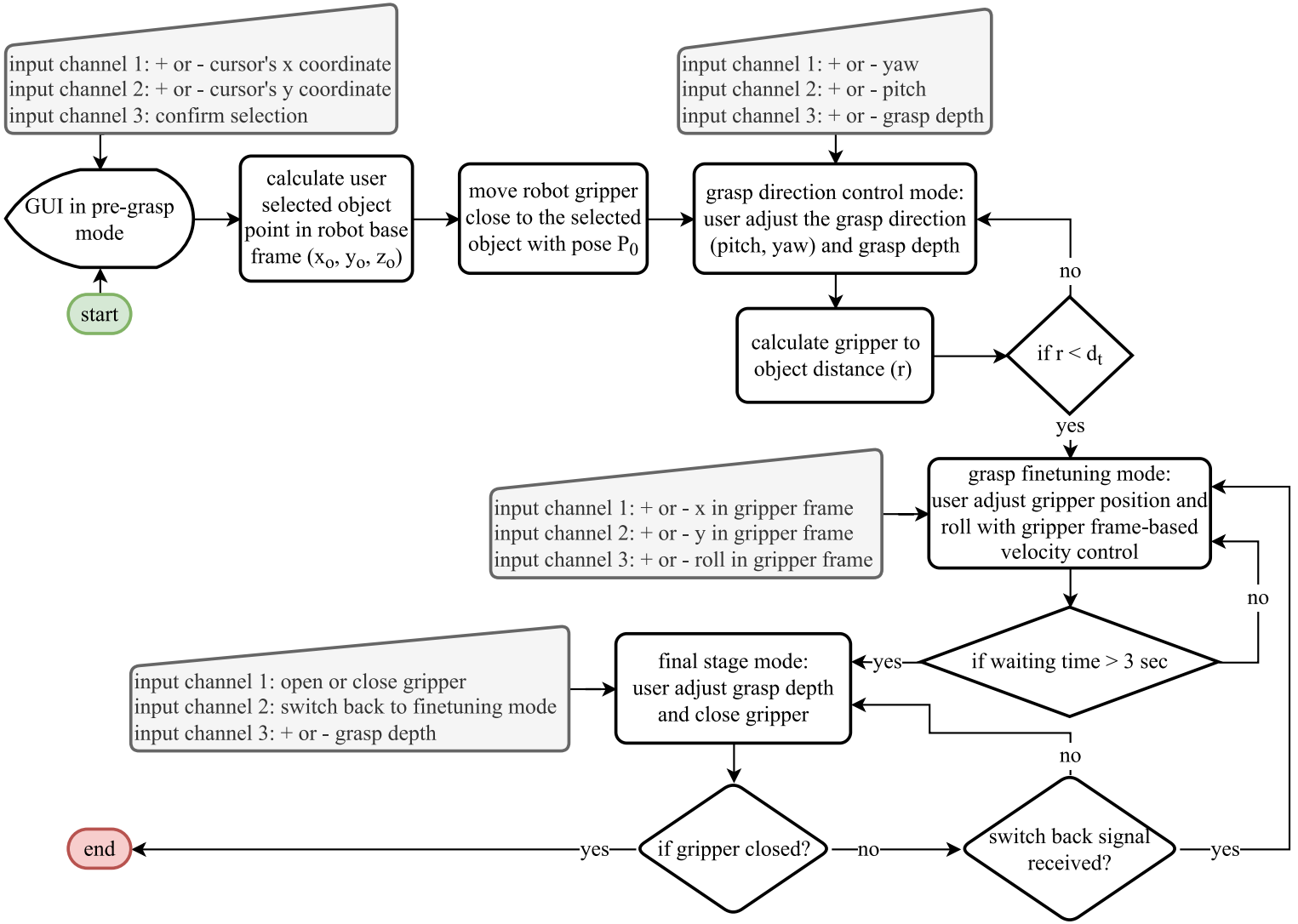


Figure 7.4 Flowchart of the assisted velocity control system.

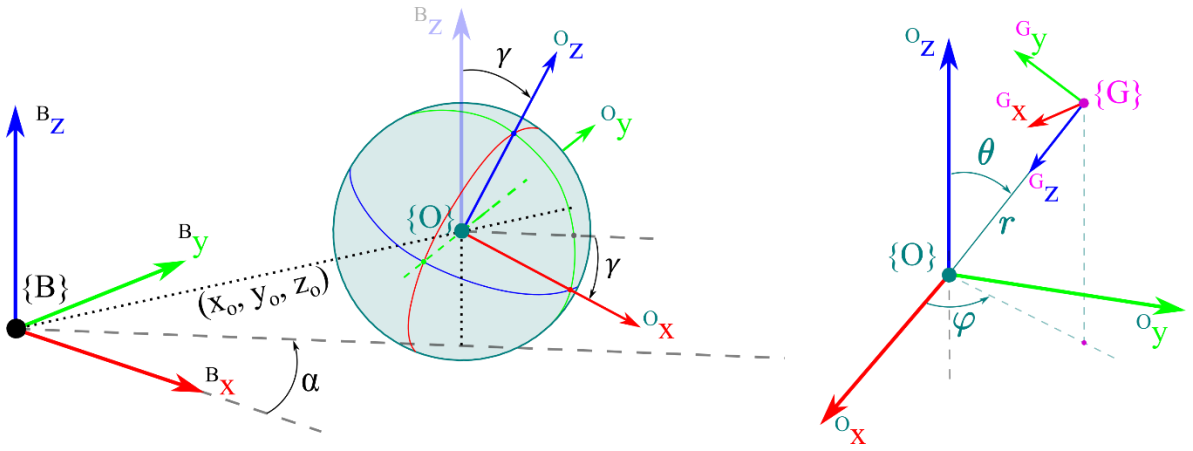


Figure 7.5 Illustration of the coordinate frames used in the assisted velocity control. Left: the object frame{O} relative to the robot base frame {B}. Right: the gripper frame {G} relative to the object frame {O}. The virtual sphere used in the grasp direction control mode is shown in the left diagram.

Right after the robot gripper moves close to the object with the pre-grasp pose, the system automatically switches to grasp direction control mode. With a traditional teleoperation method, changing the grasping direction requires the user to adjust the gripper's pitch and yaw and change the gripper's position to ensure the gripper points towards the object after the grasp direction changes. In this work, a virtual sphere is introduced to assist the user in this control mode, which allows the user to focus on controlling the grasp direction while the gripper position is adjusted automatically. As shown in Figure 7.5, the virtual sphere is centered at the user-selected object point (x_o, y_o, z_o) in the robot base frame {B}; its radius (r) is the pre-grasp distance set to 0.1 meters. When changing the grasp direction, the gripper position is constrained to stay on the surface of the virtual sphere, and the gripper's z-axis (${}^{G}z$) is forced to point towards the sphere's center. Therefore, the grasp direction and depth can be controlled by adjusting the spherical coordinates (φ, θ, r) of the gripper frame {G} in the object frame. The grasp direction control mode maps the user-controlled spherical velocity of the gripper $(\dot{\varphi}, \dot{\theta}, \dot{r})$ in the object frame {O}

to the Cartesian velocity of the gripper $(\dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z)$ in the robot base frame {B}, as formulated by equations (7.1) and (7.2).

$$ {}^{B}\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \mathrm{J} \; {}^{O}\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{r} \end{bmatrix} \tag{7.1} $$

$$ {}^{O}\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} c_1 \times 0.1\, rad/s \\ c_2 \times 0.1\, rad/s \\ c_3 \times 0.015 \quad m/s \end{bmatrix} \tag{7.2} $$

where c_1, c_2, and c_3 are the user's inputs from the input device, and the value of each input can be -1, 0, or 1. The constants in equation (7.2) are the robot gripper's angular and linear velocities selected based on the user's needs. The left side of equation (7.1) is the Cartesian velocity of the robot gripper described in the robot base frame {B}, which is also the velocity command taken by the robot control system to move the robot. The right side of the equation is the Jacobian (J) multiplied by the user input velocity vector. Therefore, the core problem of developing the grasp direction control mode is deriving the Jacobian. The linear (J_v) and angular (J_ω) velocity Jacobians are derived separately based on velocity mapping.

The first step to find the linear velocity Jacobian is to convert the gripper frame's positional coordinates in the object frame from spherical to Cartesian, as shown in equation (7.3). Then the equation for calculating the gripper's linear velocity in the object frame is simply the time derivative of equation (7.3). This chapter uses the letters *s* and *c* as the abbreviations of sine and cosine, respectively.

$$^{O}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} rc\varphi s\theta \\ rs\varphi s\theta \\ rc\theta \end{bmatrix} \tag{7.3}$$

$$^{O}\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -rs\varphi s\theta & rc\varphi c\theta & c\varphi s\theta \\ rc\varphi s\theta & rs\varphi c\theta & s\varphi s\theta \\ 0 & -rs\theta & c\theta \end{bmatrix} {}^{O}\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{r} \end{bmatrix} \tag{7.4}$$

After calculating the gripper's velocity in the object frame, the second step is to change the reference frame of this velocity to the robot base frame. As shown in equation (7.5), because the positional difference between two reference frames does not affect the velocity propagation, only the rotation matrix of the object frame relative to the robot base frame ($^{B}_{O}R$) is needed to obtain the gripper velocity in the robot base frame.

$$^{B}\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = {}^{B}_{O}R \; {}^{O}\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{7.5}$$

As shown in Figure 7.6, the intersections between the virtual sphere and the positive and negative sides of the object frame's z-axis are defined as the sphere's north and south poles, which are the singularity points of the object frame spherical coordinate system. The object frame is assigned in such a way that the singularity points are outside the robot's dexterous workspace so that the robot will not reach the singularity points. The object frame {O} is assigned by first translating the robot base frame {B} to the object by the translation vector (x_o, y_o, z_o), then rotating the translated frame about its z-axis by angle α, and finally rotating the resultant frame about its y-axis by angle γ (which is set to be 30° in this work). The first rotation is for selecting the object frame's x-axis (^{O}x) to be in the plane defined by the robot base frame's z-axis (^{B}z) and the object point vector (x_o, y_o, z_o) in the robot base frame. This selection of the x-axis will ensure

the object frame's north pole is the farthest among other choices of the x-axis after the second rotation. The second frame rotation tilts the virtual sphere's singularity points outside the robot's dexterous workspace. Figure 7.7 shows the definition of the robot's dexterous workspace in grasping. The grasp poses inside the dexterous workspace contain the most used grasp poses for the object, and the robot arm's configurations for these poses possess relatively high manipulability.

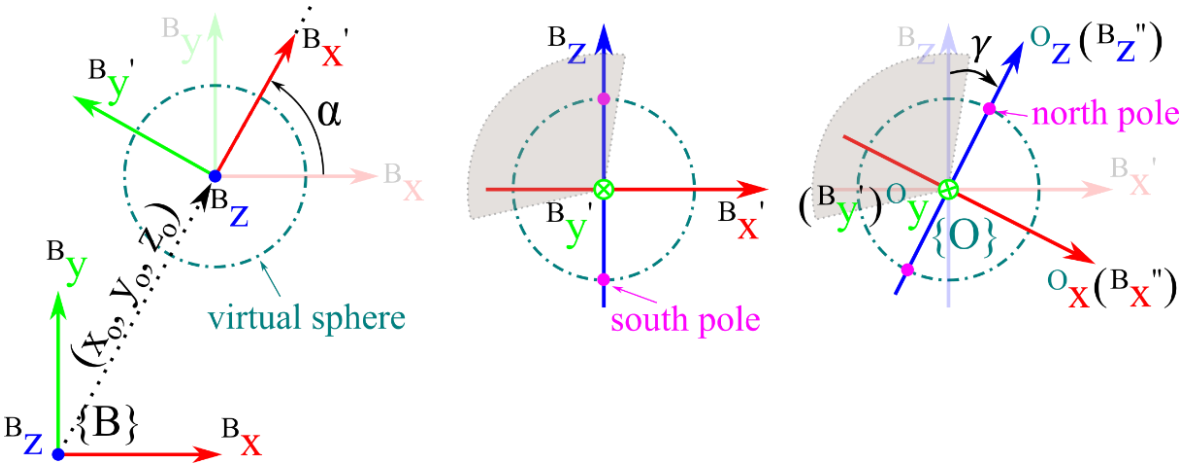


Figure 7.6 Illustration of the transformation from the robot base frame {B} to the object frame {O}. The virtual sphere for the grasp direction control and the virtual sphere's singularity points (north and south poles) are labeled.

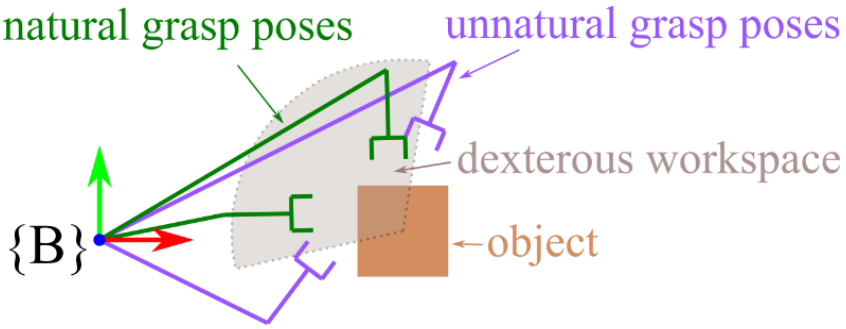


Figure 7.7 Illustration of the robot's dexterous workspace in grasping an object. The grasp poses inside the dexterous workspace are the most used natural grasp configurations and are easy for the robot to reach; the unnatural grasp poses are the opposite.

The rotation matrix of the object frame relative to the robot base frame (${}^{B}_{O}R$) is presented in equation (7.6). Based on equations (7.4) to (7.6), we can determine the gripper's linear velocity in the robot base frame as equation (7.8), from which the linear velocity Jacobian (J_v) can be extracted as shown in equation (7.9).

$$ {}^{B}_{O}R = R_z(\alpha)R_y(\gamma) = \begin{bmatrix} c\gamma c\alpha & -s\alpha & s\gamma c\alpha \\ c\gamma s\alpha & c\alpha & s\gamma s\alpha \\ -s\gamma & 0 & c\gamma \end{bmatrix} \tag{7.6} $$

$$ \begin{cases} c\alpha = \dfrac{x_o}{\sqrt{x_o^2 + y_o^2}} \\ s\alpha = \dfrac{y_o}{\sqrt{x_o^2 + y_o^2}} \end{cases} \tag{7.7} $$

$$ {}^{B}\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} c\gamma c\alpha & -s\alpha & s\gamma c\alpha \\ c\gamma s\alpha & c\alpha & s\gamma s\alpha \\ -s\gamma & 0 & c\gamma \end{bmatrix} \begin{bmatrix} -rs\varphi s\theta & rc\varphi c\theta & c\varphi s\theta \\ rc\varphi s\theta & rs\varphi c\theta & s\varphi s\theta \\ 0 & -rs\theta & c\theta \end{bmatrix} {}^{O}\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{r} \end{bmatrix} \tag{7.8} $$

$$ \mathrm{J}_v = $$

$$ \begin{bmatrix} -(c\gamma c\alpha s\varphi + s\alpha c\varphi)s\theta r & (c\gamma c\alpha c\varphi c\theta - s\alpha s\varphi c\theta - s\gamma c\alpha s\theta)r & c\gamma c\alpha c\varphi s\theta - s\alpha s\varphi s\theta + s\gamma c\alpha c\theta \\ (c\alpha c\varphi - c\gamma s\alpha s\varphi)s\theta r & (c\gamma s\alpha c\varphi c\theta + c\alpha s\varphi c\theta - s\gamma s\alpha s\theta)r & c\gamma s\alpha c\varphi s\theta + c\alpha s\varphi s\theta + s\gamma s\alpha c\theta \\ s\gamma s\varphi s\theta r & -(s\gamma c\varphi c\theta + c\gamma s\theta)r & c\gamma c\theta - s\gamma c\varphi s\theta \end{bmatrix} \tag{7.9} $$

To find the gripper's angular velocity, we must define the gripper frame first. Because the gripper must point towards the object, the gripper frame's z-axis in the object frame (${}^{O}_{G}z$) can be calculated using the gripper's position in the object frame (${}^{O}_{G}P$ in spherical coordinates) as in equation (7.10). Based on the definition of the object frame, the robot base frame's z-axis in the object frame (${}^{O}_{B}z = [-s\gamma, 0, c\gamma]^T$) is the transpose of the third row of the rotation matrix ${}^{B}_{O}R$. Let L_{2x} be the vector norm of the cross product of ${}^{O}_{B}z$ and ${}^{O}_{G}z$. The gripper's x-axis in the object frame (${}^{O}_{G}x$) is calculated using equation (7.11). Since the gripper's roll is not controlled in the direction

control mode, the selection of the gripper's x-axis can be in any direction perpendicular to the gripper's z-axis. In this work, the default direction of the gripper's x-axis is set as in equation (7.11) to make the hand camera on the top side of the gripper of our robot. The gripper's y-axis in the object frame ($^{O}_{G}y$) can be obtained as the cross product of the gripper's z-axis and x-axis in the object frame, as shown in equation (7.12).

$$^{O}_{G}z = \frac{-^{O}_{G}P}{r} = \begin{bmatrix} -c\varphi s\theta \\ -s\varphi s\theta \\ -c\theta \end{bmatrix} \tag{7.10}$$

$$^{O}_{G}x = \frac{^{O}_{B}z \times ^{O}_{G}z}{L_{2x}} = \frac{1}{L_{2x}} \begin{bmatrix} c\gamma s\varphi s\theta \\ -c\gamma c\varphi s\theta - s\gamma c\theta \\ s\gamma s\varphi s\theta \end{bmatrix} \tag{7.11}$$

$$^{O}_{G}y = ^{O}_{G}z \times ^{O}_{G}x = \frac{1}{L_{2x}} \begin{bmatrix} -s\gamma s\varphi^2 s\theta^2 - c\gamma c\varphi s\theta c\theta - s\gamma c\theta^2 \\ s\gamma s\varphi c\varphi s\theta^2 - c\gamma s\varphi s\theta c\theta \\ c\gamma c\varphi^2 s\theta^2 + s\gamma c\varphi s\theta c\theta + c\gamma s\varphi^2 s\theta^2 \end{bmatrix} \tag{7.12}$$

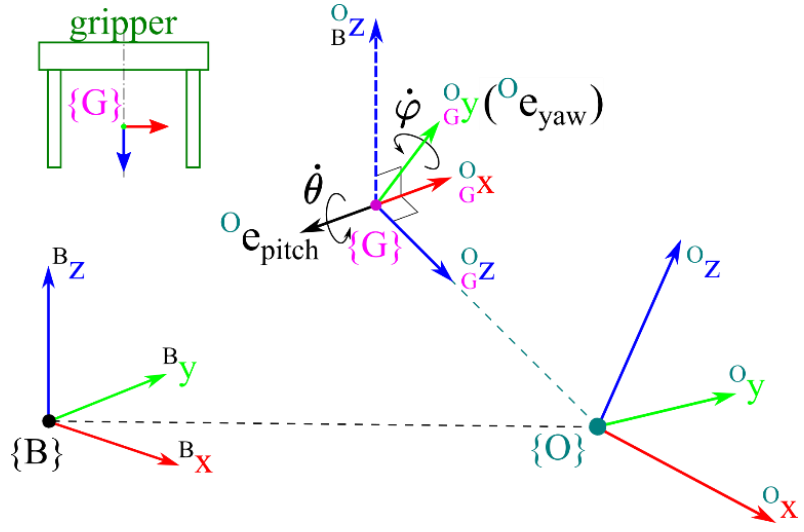


Figure 7.8 Illustration of the gripper frame and its angular velocity vectors in the object frame.

Figure 7.8 shows the defined gripper frame relative to the object frame and the gripper frame's pitch and yaw unit rotation vectors, ${}^{O}e_{yaw}$ and ${}^{O}e_{pitch}$. The gripper's yaw rotation vector is its y-axis (${}^{O}e_{yaw} = {}^{O}_{G}y$), and its pitch rotation vector is its inverted x-axis (${}^{O}e_{pitch} = -{}^{O}_{G}x$). The pitch rotation is about the inverted x-axis of the gripper because its pitch in the gripper frame and its polar angle (θ) in the object frame are negatively correlated.

Analogous to the linear velocity Jacobian deriving process, the angular velocity Jacobian can be derived from finding the gripper's angular velocity in the object frame and mapping the velocity to the robot base frame. Because the gripper's roll (rotation about its z-axis) does not affect the grasp direction, the angular velocity for grasp direction control comprises two parts, the gripper's pitch and yaw angular velocities. The gripper's angular velocity equations are shown in (7.13) to (7.15).

$$ {}^{B}\omega = {}^{B}_{O}R\ {}^{O}\omega = {}^{B}_{O}R\ {}^{O}e_{yaw}\dot{\varphi} + {}^{B}_{O}R\ {}^{O}e_{pitch}\dot{\theta} \tag{7.13} $$

$$ {}^{B}_{O}R\ {}^{O}e_{yaw} = \begin{bmatrix} c\gamma c\alpha & -s\alpha & s\gamma c\alpha \\ c\gamma s\alpha & c\alpha & s\gamma s\alpha \\ -s\gamma & 0 & c\gamma \end{bmatrix} \begin{bmatrix} -s\gamma s\varphi^2 s\theta^2 - c\gamma c\varphi s\theta c\theta - s\gamma c\theta^2 \\ s\gamma s\varphi c\varphi s\theta^2 - c\gamma s\varphi s\theta c\theta \\ c\gamma c\varphi^2 s\theta^2 + s\gamma c\varphi s\theta c\theta + c\gamma s\varphi^2 s\theta^2 \end{bmatrix} \frac{1}{L_{2x}} \tag{7.14} $$

$$ {}^{B}_{O}R\ {}^{O}e_{pitch} = \begin{bmatrix} c\gamma c\alpha & -s\alpha & s\gamma c\alpha \\ c\gamma s\alpha & c\alpha & s\gamma s\alpha \\ -s\gamma & 0 & c\gamma \end{bmatrix} \begin{bmatrix} -c\gamma s\varphi s\theta \\ c\gamma c\varphi s\theta + s\gamma c\theta \\ -s\gamma s\varphi s\theta \end{bmatrix} \frac{1}{L_{2x}} \tag{7.15} $$

Let the resultant vectors of equations (7.14) and (7.15) be represented by $\left[e_{y0}, e_{y1}, e_{y2}\right]^T$ and $\left[e_{p0}, e_{p1}, e_{p2}\right]^T$, respectively. The calculated gripper angular velocity in the robot base frame can be represented by equation (7.16), from which the angular velocity Jacobian (J_ω), as shown in equation (7.17), can be extracted. Finally, the equation (7.1) for the grasp direction control mode can be rewritten based on the linear and angular velocity Jacobians as in equation (7.18).

$$ {}^{B}\omega = {}^{B}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} = J_\omega \, {}^{O}\begin{bmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}e_{y0}\\ e_{y1}\\ e_{y2}\end{bmatrix}\dot{\varphi} + \begin{bmatrix}e_{p0}\\ e_{p1}\\ e_{p2}\end{bmatrix}\dot{\theta} \tag{7.16} $$

$$ J_\omega = \begin{bmatrix}e_{y0} & e_{p0} & 0\\ e_{y1} & e_{p1} & 0\\ e_{y2} & e_{p2} & 0\end{bmatrix} \tag{7.17} $$

$$ {}^{B}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \omega_x\\ \omega_y\\ \omega_z\end{bmatrix} = \begin{bmatrix}\mathrm{J}_v\\ \mathrm{J}_\omega\end{bmatrix} {}^{O}\begin{bmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{r}\end{bmatrix} \tag{7.18} $$

In the grasp direction control mode, the user first adjusts the gripper's grasp direction using the first two input channels, then moves the gripper close to the object along the grasp direction using the third input channel. When the distance between the gripper frame and the object frame (the virtual sphere's radius r) is smaller than a threshold (0.04m is used in this work), the system automatically switches to the grasp pose fine-tuning mode. In this fine-tuning mode, the three input channels switch to control the gripper's x, y, and roll in the gripper frame so that the user can fine-tune the planar position and orientation of the gripper along the plane perpendicular to the gripper's z-axis. The user's input and the corresponding velocity command sent to the robot controller are shown in equation (7.19).

$$ {}^{G}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \omega_x\\ \omega_y\\ \omega_z\end{bmatrix} = \begin{bmatrix}c_1 \times 0.01 & m/s\\ c_2 \times 0.01 & m/s\\ 0 & \\ 0 & \\ 0 & \\ c_3 \times 0.07 & rad/s\end{bmatrix} \tag{7.19} $$

When the grasp pose is tuned to the user's satisfaction, the user stops operating the robot and waits for the system to switch to the final stage mode. After the finetuning mode, if there is no user input for 3 seconds, the system automatically switches to the final stage mode. In this mode, the user can use the third input channel to finetune the grasp depth, the second input channel to return to the previous control mode, and the first to open or close the gripper.

Chapter 8: Experiments and Results

Three experiments were conducted in this dissertation to test the object movement-based grasp quality measure and the developed HitL robot grasping system. The object movement-based grasp quality measure was tested with a real robot grasping experiment and an image-based grasp planning experiment. The robot grasping experiment was conducted to show if the grasp quality scores calculated using the grasp quality measure can match the real robot grasping results. The image-based grasp planning experiment was conducted to compare the object movement-based quality measure with other grasp quality measures. The HitL robot grasping system was tested against a baseline grasping system in another robot grasping experiment.

8.1 Grasp Quality Measure Robot Grasping Experiment and Results

8.1.1 Experiment Setup

The Baxter robot [59] from Rethink Robotics was used as the test platform for this robot grasping experiment. The programs were written in Python and performed on a laptop PC running Ubuntu 18.04 with a 2.2 GHz Intel Core i7-8750H CPU, 8 GB of RAM, and an NVIDIA GeForce GTX 1060 graphics card. The graphics card was only used for computing the objects' silhouettes, and all other computations were performed on the CPU. A parallel jaw gripper with a pair of narrow fingers (1.3 cm wide) was used, and the gripper fingers' rubber contact surfaces were covered with scotch tape to reduce friction so that slipping is more likely to happen when the grasp pose is prone to slippage. Figure 8.1 shows the robot hand and the ten objects used in the grasping experiment.

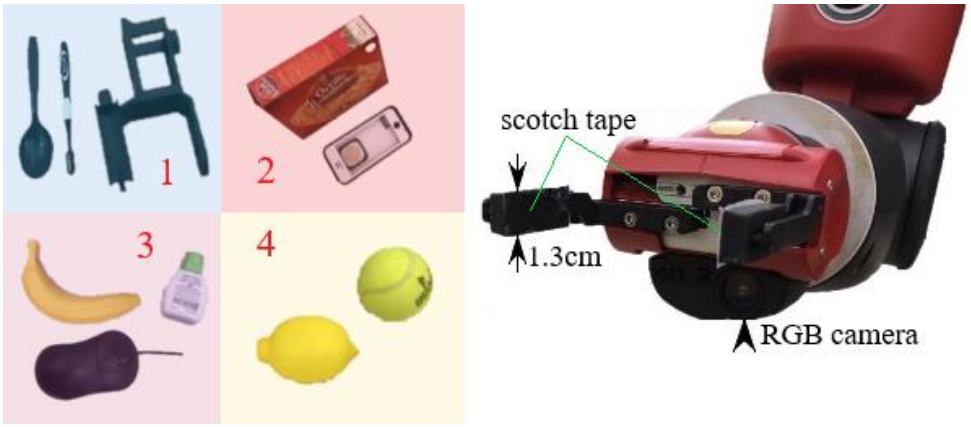


Figure 8.1 The robot gripper and the objects used in the robot grasping experiment. The numbers indicate the categories of objects, and categories 1 and 2 both have flat contact surfaces. However, the graspable parts of category-1 objects are much thinner than category-2 objects. Category-4 objects have curved contact surfaces, and category-3 objects have flat and curved contact surfaces.

This experiment examines how the calculated grasp quality score relates to the actual robot grasping performance. In the experiment, an eye-to-hand RGB-D camera (Intel Realsense L515) was used to locate the object. The gripper pitch and yaw were set as -90 and 0 degrees, respectively, to form top-down grasps. Then, an RGB eye-in-hand camera (Baxter hand camera) was used to take a closer shot at the object from the top. Facebook Detectron2 [38] was used to extract the object's silhouette from the closely shot object image. Assuming the gripper was at the object's location, the gripper's projection in the object silhouette image can be calculated using the calibrated camera intrinsic and the object's distance to the camera. With the object and gripper projections, grasp pose candidates were randomly generated within the object's bounding box. After each grasp candidate was generated, it was evaluated by the object movement-based grasp quality measure to check if it meets the quality score requirement for robot execution. This heuristic searching stops when the grasp pose of the desired quality score is found. Randomly

generated grasp poses help prove the quality measure's robustness since getting the desired pose requires hundreds and thousands of candidates to be evaluated.

To test the whole grasp score value domain [0, 1], we separated the grasp score into ten score levels, 0-0.1, 0.1-0.2, ..., 0.9-1. Ten grasp trials were performed for each object in each score level, resulting in 1000 grasps for ten objects and ten score levels. The number of evaluated grasp candidates, the time used in grasp evaluation, and the robot grasp outcome were recorded for each grasp trial. The robot grasp outcome includes (1) a binary term that indicates if the grasp is successful and (2) two hand camera images (BA images) taken right before and after the gripper closes, which capture the object's movement during the gripper closing. The grasp is recorded as successful if the robot can lift the object for 2 seconds and put it back without visible slippage. In the BA images, the object's rotation and translation in the gripper closing direction were measured as the key lines' orientation and the key points' position difference. We drew the key lines using the object's appearance features that are visible in both BA images. We also drew the position key point as the object's center on the gripper's center grasp line.

8.1.2 Results and Discussion

In terms of the computational cost, the recorded average grasp generation time for the 1000 grasps is 0.534 seconds. 261,868 grasp candidates were evaluated to generate those grasps; the average grasp evaluation time is 2.04 milliseconds, with a standard deviation of 0.264 milliseconds. In terms of the accuracy of the quality measure, Figure 8.2 presents the results of the robot grasping experiment. In Figure 8.2, the x-axis is the average grasp score, the left y-axis is the success rate; the right y-axis is the normalized object movement score. Because the objects' movements in the failed grasp trials were unpredictable and not helpful in evaluating our method, only the objects' movements in succeeded grasp trials were measured to reveal the quality

difference among those successful grasp poses. The object movement score was calculated as the average of the normalized object rotation and translation (in the gripper closing direction). The thresholds for normalizing the object translation and rotation were 100 pixels and 60 degrees, respectively. The thin non-solid lines show each object's average success rate at each score level. The thick solid line shows the average success rate, and the bars show the average object movement of all objects at each score level. To help interpret the results, we classified the test objects into four categories based on their geometrical properties. As shown in Figure 8.1, objects in categories 1 and 2 both have flat contact surfaces. However, the graspable parts of category-1 objects are much thinner than category-2 objects. Category-4 objects have curved contact surfaces, and category-3 objects have flat and curved contact surfaces.

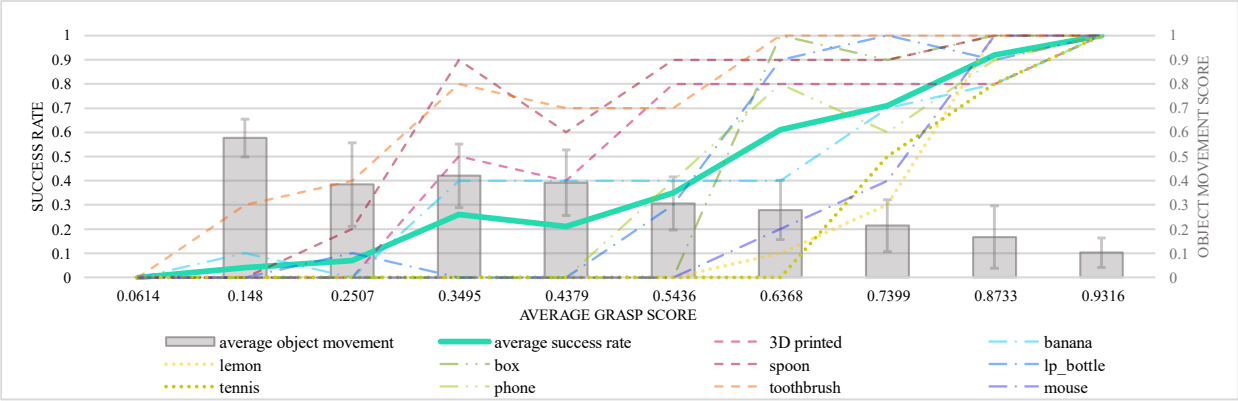


Figure 8.2 The success rate and the object movement score of the robot grasping experiment vs. the grasp score.

In terms of the object-wise success rate, the objects in the same category tend to have similar success rates under the same score level. Also, category-1 and category-4 objects have the highest and lowest overall grasp success rates, respectively. The category-1 objects are long and thin and have flat contact surfaces in the projection, making them the easiest for parallel-jaw

grippers to grasp, while category-4 objects are the hardest to grasp because they are round and our gripper fingers are narrow. The category-1 objects and the banana have relatively high success rates even when the grasp qualities are low. This result does not disprove our quality measure because the object movements in those grasps are very high. Therefore, even though the grasps were successful, they were evaluated as low quality because they moved the objects a lot.

Despite the differences between different objects' success rates in the medium quality score levels, all objects' success rates merge to the same points at the highest and lowest score levels, which indicates that our quality measure can distinguish good and bad grasps regardless of the object type. From the overview of the grasp score and quality relationship, we can conclude that when the grasp quality score increases, the average grasp success rate increases and the average object movement decreases accordingly, which proves that the calculated grasp quality scores can indeed predict the actual qualities of the grasp poses. In addition, the 100% success rate in grasping the ten objects 100 times shows the high accuracy of our quality measure.

8.2 Grasp Quality Measure Comparison Experiment and Results

8.2.1 Experiment Setup

Four grasp planning systems were implemented for this comparison, which include two analytical systems based on our quality measure and the Q-measure [40], and two deep learning-based systems, the FC-GQCNN [20] and the GGCNN [19]. For the grasp planning system using our grasp quality measure, a normally distributed random search around the object's geometric center was used to find a high-quality grasp pose (quality score > 0.95). If no such high-quality pose was found, then a uniformly distributed small region random search around the grasp pose with the maximum score was performed to find a local pseudo-optimum. The same searching method was used for the Q-measure-based grasp planning system, except it does not stop searching

at a quality score threshold, instead, it keeps searching until it finds the global pseudo-optimum. The values 0.8 and 1 were used as the friction coefficient and the torque scaling factor, respectively, in evaluating grasp poses' Q-measure scores. The deep learning-based grasp planning systems are end-to-end networks that do not require a separate grasp pose candidates sampling system, which makes them more efficient than analytical systems. However, they are often much harder to implement than the analytical ones in real-world applications since they usually need to be re-trained to adapt to different working environment setups.

For this comparison, the pre-trained models named FC-GQCNN-4.0-PJ and GGCNN were used for the FC-GQCNN and the GGCNN methods, respectively. This experiment used the two networks' training datasets as the test inputs. These test inputs were comprised of 100 single-object images extracted from the DexNet 2.0 grasping dataset (synthetic) [18] and the Cornell grasping dataset (real-world) [17]. The 100 object images include 50 from the DexNet dataset and 50 from the Cornell grasping dataset. Only the images with good mask detection results were selected from the Cornell grasping dataset to rule out the impact of flawed inputs on the grasp planning results. The two analytical grasp planning systems were tested with all 100 images, and the FC-GQCNN and the GGCNN methods were tested with the DexNet subset and the Cornell subset, respectively. Some example objects from this test dataset are shown in Figure 8.3.

Figure 8.3 Sample objects from the Cornell and DexNet grasping datasets used in the method comparison test.

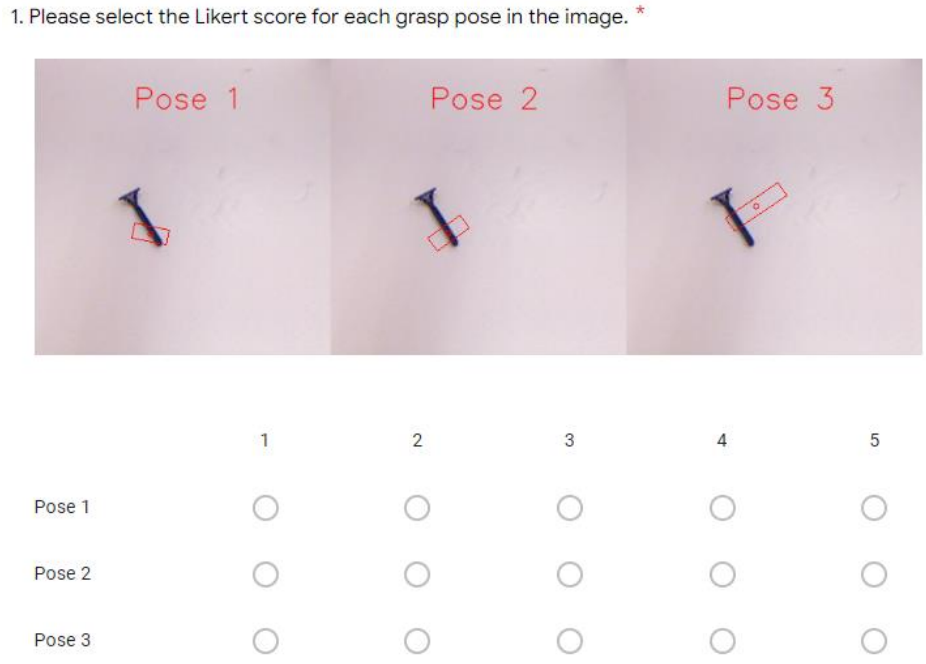


Figure 8.4 Survey question example: grasp detection results comparison based on the Cornell grasping dataset.

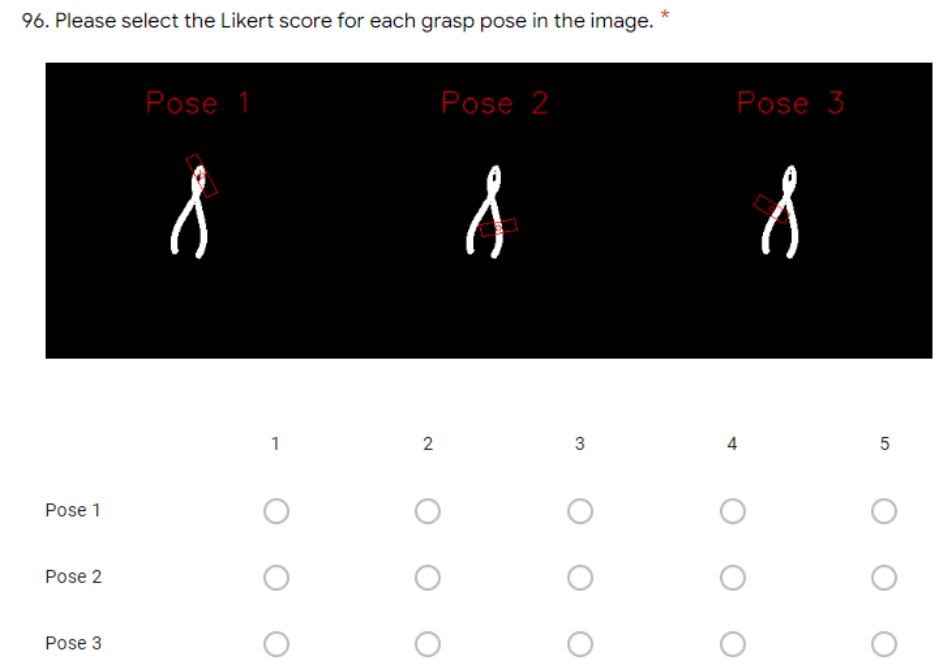


Figure 8.5 Survey question example: grasp detection results comparison based on the DexNet grasping dataset.

After obtaining the grasp planning results, an anonymous online survey (USF IRB# Pro00040871) of 100 questions (one for each input image) was created. For each question, the participants see three copies of the input image, and each image shows one grasp planning result (drawn on the image as a rectangle) from the three grasp planning methods tested on that image, as shown in Figures 8.4 and 8.5. Then the participants are asked to compare and evaluate the quality of each grasp pose (without knowing which pose is from which planning method) based on a 5-point Likert scale. The participants were instructed to score the grasp poses using the following criteria:

1. The worst case. The robot gripper fingers will collide with the object, or the gripper will miss the object.

2. Better than 1. The grasp does not miss the object, and there is no collision, but the robot still has a low chance of picking up the object.
3. Better than 2. The robot has a very high chance of picking up the object. However, there will be a large amount of object movement.
4. Better than 3. The robot can pick up the object, and there is only a small amount of object movement.
5. The best case. The robot can pick up the object, and there is almost no object movement.

This survey uses human evaluation as a baseline to compare the results of different grasp planning methods, which can indicate the effectiveness of the corresponding grasp quality measures. Figures 8.4 and 8.5 are two example questions from the survey.

8.2.2 Results and Discussion

Ten subjects participated in the anonymous survey for the method comparison experiment. Figures 8.6 and 8.7 show the percentage of the planned grasp poses based on their human-evaluated Likert score. For each dataset, the resulting grasp poses percentages were calculated based on 500 human evaluations (50 instances times 10 participants). ANOVA tests revealed that the average quality scores of the grasp poses generated by the three compared methods (in each sub-test group) are significantly different. For the sub-test group of GGCNN, Q-measure, and our method, $F(2,1497)=672.745$, $p<0.001$. For the sub-test group of GQCNN, Q-measure, and our method, $F(2,1497)=17.645$, $p<0.001$. Post-hoc analyses revealed the following results:

1. In the Cornell grasping dataset-based grasp planning test, the average quality of grasps generated by our quality measure-based grasp planner (4.244 ± 0.894) was significantly higher than the ones generated by the GGCNN-based grasp planner (1.936 ± 1.515, $p<0.001$), and the ones generated by the Q-measure-based grasp planner (3.914 ± 1.065, $p<0.001$).

2. In the DexNet grasping dataset-based grasp planning test, the average quality of grasps generated by our quality measure-based grasp planner (4.042±0.994) was significantly higher than the ones generated by the GQCNN-based grasp planner (3.638±1.454, $p<0.001$). However, there is no significant difference between our quality measure and the Q-measure-based (3.924±1.221, $p=0.0765$) grasp planners.

These results show that the grasp planning system using our quality measure generates significantly more top-quality grasp poses (scored as 5) and fewer poor-quality grasp poses (scored as 1 or 2). In addition, both analytical grasp planning systems work almost consistently across datasets. There was a non-negligible percentage reduction in the generated top-quality grasp poses for the DexNet dataset objects, which is as expected since the objects in this dataset are significantly more complex than those in the Cornell dataset. Despite the performance difference between the two grasp planning networks, both generated the largest number of poor-quality poses and the smallest number of top-quality poses.

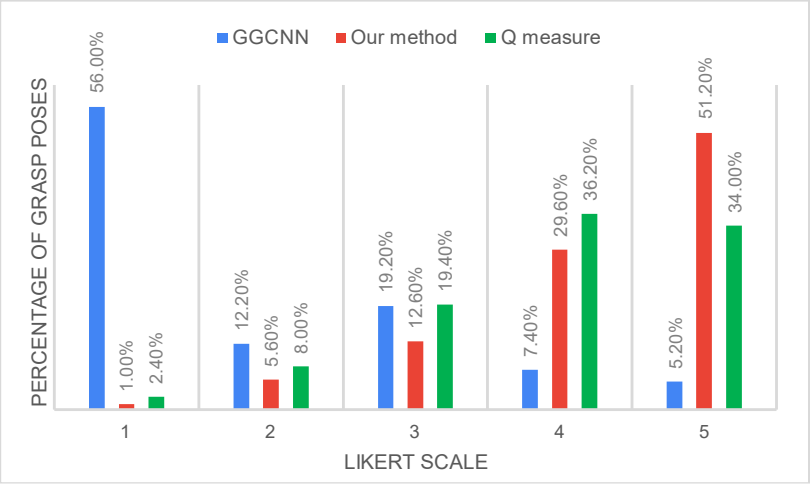


Figure 8.6 The results of grasp planning on the Cornell sub-dataset using the GGCNN, our quality measure, and the Q-measure. 1 and 5 indicate the worst and best quality grasp poses, respectively.

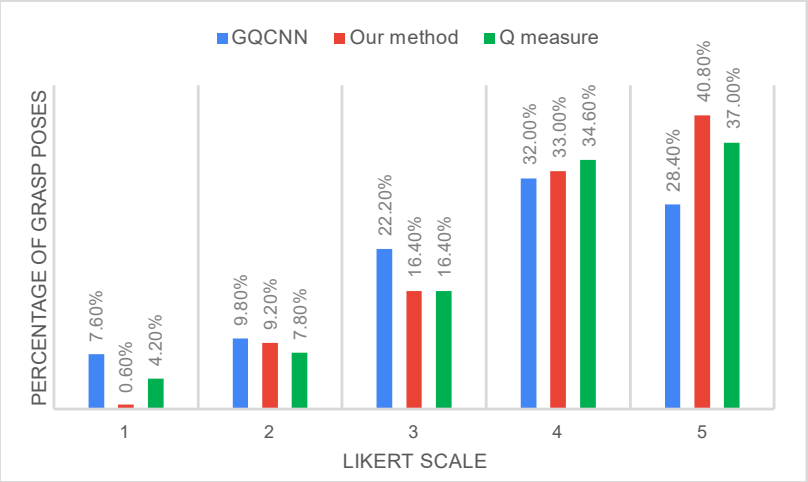


Figure 8.7 The results of grasp planning on the DexNet sub-dataset using the FCGQCNN, our quality measure, and the Q-measure. 1 and 5 indicate the worst and best quality grasp poses, respectively.

As the Q-measure-based grasp planning system has similar performance to our quality measure-based grasp planning system, a more detailed individual case comparison was conducted to highlight the difference between the two quality measures. As shown in Figure 8.8, four grasp poses were evaluated by both quality measures, and through this comparison, we can find the following differences:

1. The Q-measure cannot identify the quality difference between grasps 1 and 2 because it evaluates grasp quality by grasp contact wrenches and grasp 1 and 2 have nearly identical contact wrenches.
2. Comparing the two same-quality grasps, 1 and 3, it is apparent that the Q-measure is more sensitive to flawed input. Because of the imperfect object silhouette, our quality evaluation has a score change of 10.4%, while the Q-measure has a score change of 79.8%.
3. The Q-measure cannot evaluate grasp poses with contact forces outside of the predetermined contact friction cone (grasp 4), while our quality measure does not have this limitation.

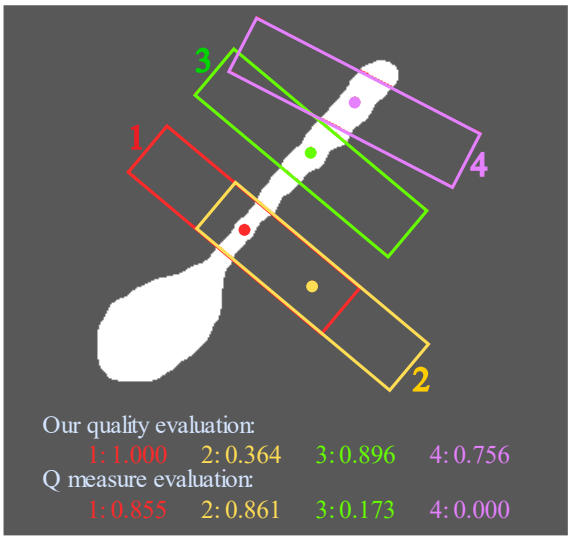


Figure 8.8 Comparison between our grasp quality measure and the Q-measure in evaluating grasp poses. The numbers 1 to 4 are the index of the grasp poses, and the numbers after the index numbers are the quality scores of the corresponding grasp poses.

8.3 HitL Robot Grasping System Experiment and Results

8.3.1 Experiment Setup

Our HitL robot grasping system presented in this dissertation was developed by adding a grasp refinement module to a state-of-the-art deep learning-based grasp planning module (Chapter 5) to form a hybrid system for better grasping performance. Using the deep learning-based grasp planning module as the baseline grasping system, we tested our grasping system's performance through a real robot grasping experiment. This robot grasping experiment was performed on ten common household objects with localization cards. The localization cards were attached to the objects to measure the objects' movements. As shown in Figure 8.9, a card with four holes in its four corners was attached to the bottom of every test object. The four holes of the card are used to draw the object's localization points or recover the object's pose using the localization points. The objects are approximately aligned to the centers of their localization cards.

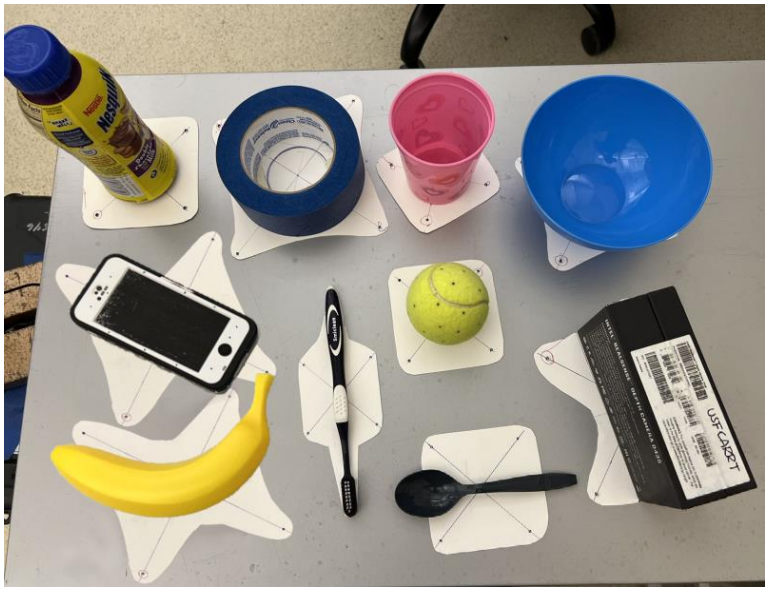


Figure 8.9 The objects used in the robot grasping experiment for testing the grasp refinement system.

With the localization cards, the objects' translation (x, y) and rotation (θ_z) in the horizontal plane can be measured, as shown in Figure 8.10 (left side). The red and green localization points represent the object pose before and after the object is grasped. The object translation is measured as the travel distance of the center of the object's localization card. The object rotation is measured as the rotation angle of the object's localization card. Because the object's movements (z, θ_x, θ_y) in the vertical direction are difficult to measure, the maximum gap between the corners of the object's localization card and the table surface was used to approximate the object's movement in the vertical direction. In this dissertation, the measurement used to approximate the object's movements in the vertical direction is referred to as the object's tilt. Figure 8.10 (right side) shows how a grasped object's tilt was measured.

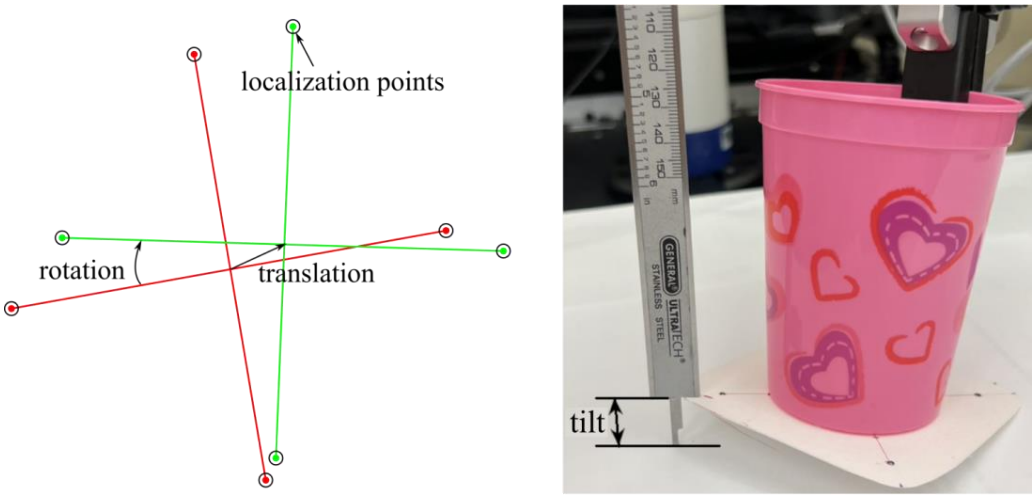


Figure 8.10 Illustration of the object movement measurements. Left: the object translation and rotation measured using the object localization points. Right: the object tilt is measured as the maximum elevation of the localization card's corners.

In terms of the test platform, a wheelchair-mounted robotic arm, as shown in Figure 8.11, was used. The robot arm is the 7-DoF Kinova Gen3 [60], and the robot gripper is the ROBOTIQ 2F-85 [50]. An Intel RealSense D435i RGB-D camera [61] was used to replace the robot arm's onboard camera for better depth sensing. The programs were written in Python, and two desktop computers were used to run the programs. Each computer has the Ubuntu 18.04 operating system, a 2.2 GHz Intel Core i7-8750H CPU, 16 GB of RAM, and an NVIDIA GeForce GTX 1080 graphics card. While most of the programs run on one computer, the additional computer is used to run the object segmentation neural network. Because combining the two neural networks for object segmentation and grasp detection is out of the scope of this work, two computers are used in the experiment so that the two neural networks can be set up separately for convenience. The number of computers used does not affect the results of the experiment.

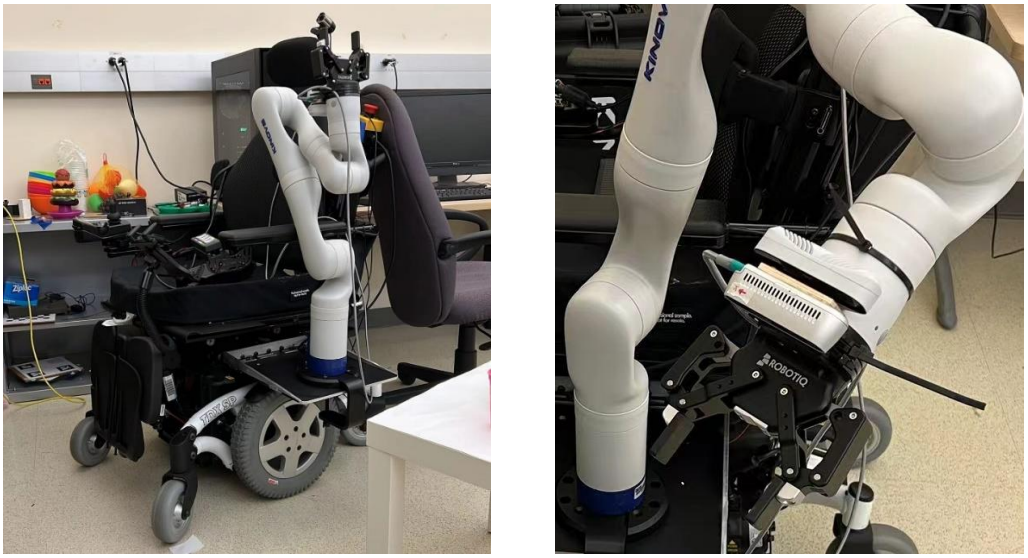


Figure 8.11 The robot platform used in the grasping experiment for testing the performance of the grasp refinement system.

In this experiment, ten grasp trials were performed on each object, and the cup and bottle each had ten additional grasp trials to increase the variety of the tested grasp direction. The experimental procedures of each grasp trial are the following:

1. Place a clean sheet of paper within the robot's workspace and tape it to the tabletop.
2. Place the test object on the paper and draw the object's initial localization points.
3. Scan the scene with two predefined view poses using the robot's hand camera.
4. Use the GUI of the grasping system to select the object to be grasped.
5. The baseline grasping system generates a baseline grasp pose, and the time used in generating the grasp pose is recorded (All the time measurements are performed using the Python time function).
6. If the baseline grasping system succeeds, then the baseline grasp pose is refined by the grasp refinement module to get the grasp pose generated by our grasping system. The time for the

grasp refinement process is recorded and added to the baseline grasp generation time to get our grasping system's total grasp planning time.

7. If the baseline grasping system fails, it indicates that the selected object has a small height and cannot be detected by the grasp detection network. Because the object's height is low, it is best to grasp it from the top. Therefore, the robot moves to a top-down view of the object and calls the object segmentation network to get the object's silhouette. Then our grasping system can use the object silhouette to generate a top-down grasp pose. The time used in the robot changing view pose, object segmentation, and grasp generation is added to the time used in the failed grasp detection process to obtain our grasp system's total grasp planning time.
8. The robot performs the grasp generated by our grasping system, and right after the gripper is closed, the object's localization points are drawn on the paper. Then the robot lifts the object for 3 seconds, and if the object does not fall, the measured grasp result is recorded as success. If a grasp's measured result is a success, but the grasp pose is extremely unstable from observation, then the grasp's observed result is recorded as a failure. Figure 8.12 shows two examples of successful (measured) unstable grasps.
9. Place the object back to the initial pose, as before grasping, by aligning the holes on the object's localization card to the initial localization points on the paper.
10. The robot performs the baseline grasp if there is a baseline grasp pose. The object's localization points and the grasp result are recorded the same way as in step 8.

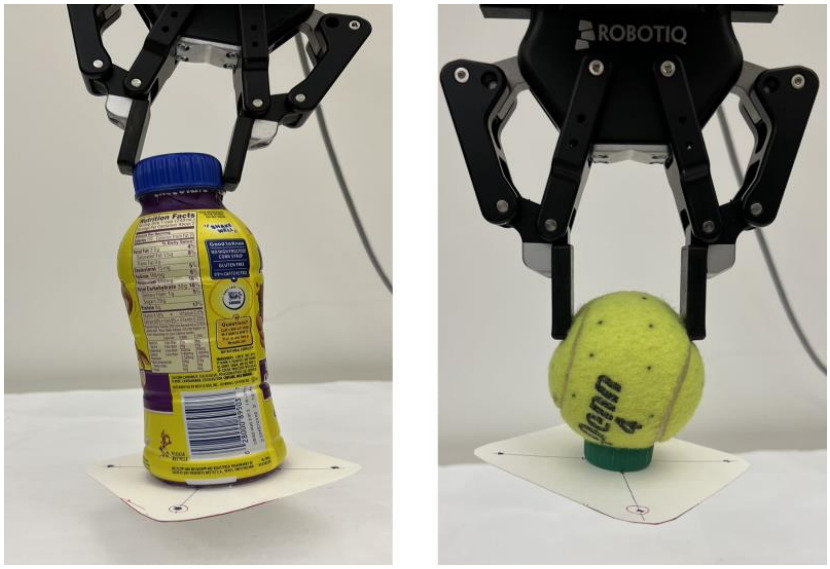


Figure 8.12 Examples of unstable robot grasps that are measured as successful based on the commonly used metric.

8.3.2 Results and Discussion

Twelve sets of grasping tests were conducted in this experiment, each containing ten grasp trials. In each grasp trial, two grasp attempts were performed with the grasp poses generated by the baseline and our grasping system. For the baseline grasping system, the grasping test sets for the spoon, toothbrush, and cell phone (group-1) yielded no grasping results because the baseline system failed to generate any grasp pose for those objects. Therefore, the experimental results are separated into two groups based on whether the baseline system can generate grasp poses for the tested object. Group-1 results contain the grasping results of the spoon, toothbrush, and cell phone. These objects are of low height (less than 2 cm), which makes them indistinguishable from the background table; therefore, they cannot be detected by the baseline grasping system, and only our grasping system performed grasping trials for these objects.

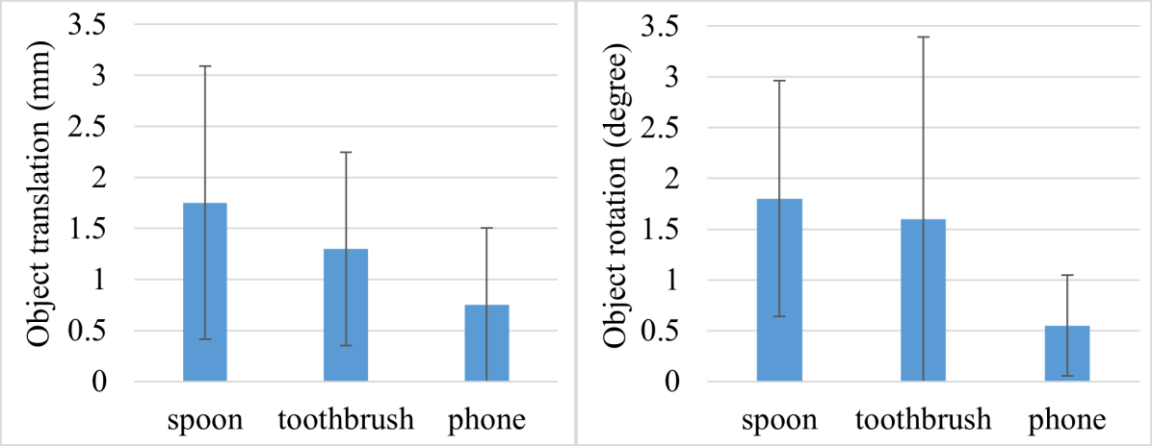


Figure 8.13 The average object movements of group-1 objects (no grasp pose detected by the baseline system) in the robot grasping experiment. The error bars present the sample standard deviations. The left is the average object translation chart, and the right is the average object rotation chart.

Figure 8.13 presents the average object translation and rotation of group-1 objects with sample standard deviations. The object tilt measurements are not presented because they are all zeros. Our grasping system achieved a 100% grasp success rate for these three objects, and the object movements that occurred in these grasping trials were very small. The average object translation was between 0.75 and 1.75 millimeters, and the maximum standard deviation was 1.34 millimeters. The average object rotation was between 0.55 and 1.8 degrees, and the maximum standard deviation was 1.8 degrees. The average grasp planning time was 6.67 seconds which is composed of the average time for the robot to change view pose (6.15 seconds), the average time for object segmentation (0.27 seconds), and the average grasp pose generation time (0.25 seconds). This time is acceptable for real-life applications, and because most of the time was spent on changing the view pose, this time can be shortened by increasing the robot's speed (within a safe range). These results proved that our robot grasping system could work for a wider range of objects than the baseline grasping system, and our grasp refinement system can generate high-quality

grasp poses for these objects. Especially considering that the finger length of the robot gripper used in this experiment changes with the gripper's open width, this increases the difficulty in performing successful top-down grasps on low-height objects. In all the grasp trials, the grasp pose generated by our grasping system had never missed the object to be grasped or caused any collision, which proves the effectiveness of the grasp depth refinement in our grasping system.

Group-2 objects include the cup, banana, bottle, bowl, box, tape, and ball, which is the group of objects that resulted in a successful grasp pose generation using the baseline system. There were nine sets of experiments conducted for these seven objects. Two additional experiment sets were added to increase the variety of the tested grasp directions for the cup and bottle. The terms cup-top, cup-side, bottle-top, and bottle-side represent the grasping test sets for the cup and bottle for top-down and side grasp direction categories, respectively. In this group of results, our grasping system can be compared with the baseline directly. In terms of computational cost, our grasping system requires more computational power since there are more processes for improving the grasp quality. The average computational time of our grasping system in each set of tests was 0.36 to 1.35 seconds, more than the baseline system, which had an average computing time of 0.57 seconds. In addition, our grasping system needs one extra view of the scene, which takes time for the robot to reach the view pose. The extra time cost on this depends on the robot's speed, and in this experiment, our robot needed about 7 seconds.

In terms of the grasp success rate of the 90 grasp trials performed on the group-2 objects, our grasping system achieved a 100% measured grasp success rate and a 99% observed grasp success rate, while the baseline system achieved a 76% measured grasp success rate and a 66% observed grasp success rate. The baseline grasping system had a low grasp success rate because many of the failed cases were due to insufficient grasp depth, which can be fixed by adding a

certain amount of gripper movement in the grasp direction to form a more stable grasp. If the grasp trials that failed because of insufficient grasp depth were counted as successful grasps, then the baseline system's grasp success rate can be increased to 94% (measured) and 84% (observed). Comparing the success rate for the baseline trials, 94% to 76% (or 84% to 66%), we can see that the grasp depths of the grasp poses generated by the baseline grasping system are unreliable. While in our grasping system, we addressed this problem using grasp depth refinement. Comparing the two success rates of our grasping system (100%, 99%) with the two ideal grasp success rates of the baseline system (94%, 84%), the 6% of measured grasp success rate increase and the 15% of observed grasp success rate increase prove that our grasping system can generate better quality grasp poses. The difference between the measured and the observed grasp success rates for our grasping system (1%) is much smaller than that for the baseline system (10%), which reveals that in the grasp trials that were measured as successful, the unstable grasp poses generated by our grasping system were significantly less than the ones generated by the baseline system.

While the grasp success/failure is a binary measure of the grasp quality, which cannot show the actual grasp quality differences between grasp poses, the object movements in grasping were recorded as a continuous measure to compare the grasp quality. In this experiment, the measured object movements included three types, the object's tilt, planar translation, and planar rotation. To thoroughly examine the measured object movements in the experiment, the object movement results are discussed case-wise and object-wise. The case-wise discussion analyzes the object movements based on each grasp trial, while the object-wise discussion analyzes the average object movements of each grasped object.

From the case-wise view, in the 90 grasp trials, there were three trials (3.33%) in which our grasping system's grasp poses caused more object movements (for all three movement types)

than the corresponding grasp poses of the baseline system. There were 67 trials (74.5%) in which our grasping system's grasp poses caused fewer object movements (for all three movement types) than the corresponding grasp poses of the baseline system. There were 20 trials (22.2%) in which our grasping system's grasp poses caused fewer object movements (for two movement types) than the corresponding grasp poses of the baseline system. In the cases that our grasping system caused more object movements, the largest movement differences in the three types of object movements are 14 mm, 2.5 mm, and 3 degrees for the object's tilt, translation, and rotation, respectively. In the cases where the baseline system caused more object movements, the largest movement differences in the three types of object movements are 19 mm, 10.5 mm, and 12.5 degrees for the object's tilt, translation, and rotation, respectively. These results show that in the cases where our grasping system caused more object movements than the baseline system (3.33%), the object movement differences were small, which means the actual qualities of the grasps generated by the two systems were close. While in the cases where the baseline grasping system caused more object movements than our system (74.5%), the object movement differences were large, which means that the actual qualities of the grasps generated by our grasping system were better than the ones generated by the baseline system.

From the object-wise view, the group-2 object's average translation, tilt, and rotation measured from the refined grasp poses (2.73±2.03 mm, 1.72±4.85 mm, and 1.34±1.82 degrees) were significantly smaller than the ones measured from the initial grasp poses (6.47±4.2 mm, $p<0.001$, 4.3±8.09 mm, $p<0.025$, and 3.75±4.02 degrees, $p<0.001$), which proves that our grasp pose refinement method-based robot grasping system can generate grasp poses that cause less (statistically significant) object movements than the tested state-of-the-art deep learning-based grasping system. The average object translation, rotation, and tilt for each test set are presented in

Figures 8.14 to 8.16. Figure 8.14 shows that our grasping system has less average object translation for every test set except for the box test set, which has roughly the same object translation for both grasping systems. However, in Figure 8.13, our grasping system caused much less object rotation in grasping the box. Figure 8.14 shows that both grasping systems caused zero object tilt in grasping the box. Therefore, after combining the three types of object movements in the box grasping test, our grasping system caused fewer overall object movements. Analogously, the overall object movement of other tested objects can be analyzed. We can see that the grasp poses generated by our grasping system caused fewer object movements than the ones generated by the baseline grasping system. In summary, the results of the grasp success rate comparison and the case-wise and object-wise object movement comparisons all prove that our grasping system can generate more accurate grasp poses that cause fewer object movements than the baseline system.

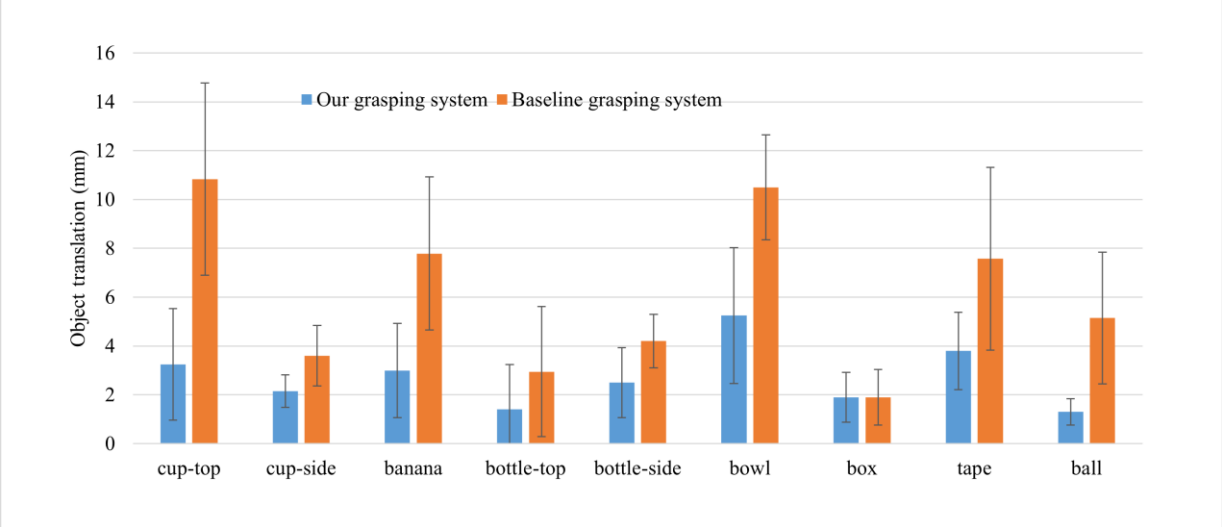


Figure 8.14 The average object translation of each set of the grasping tests for the baseline and our grasping systems. The error bars present the sample standard deviations.

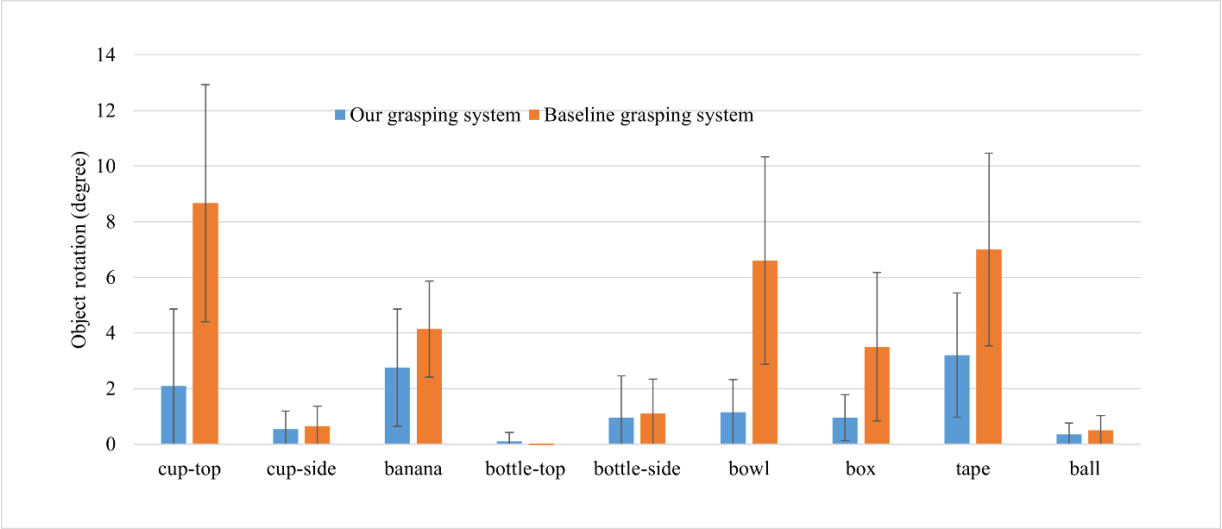


Figure 8.15 The average object rotation of each set of the grasping tests for the baseline and our grasping systems. The error bars present the sample standard deviations.

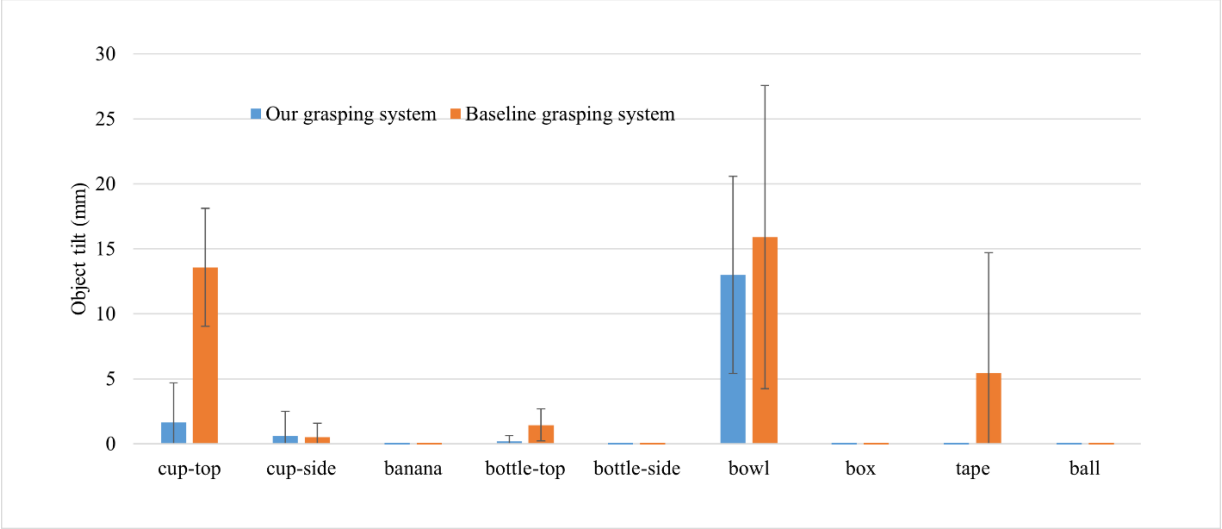


Figure 8.16 The average object tilt of each set of the grasping tests for the baseline and our grasping systems. The error bars present the sample standard deviations.

Chapter 9: Conclusions and Future Work

This dissertation presents the detailed design and experimental results of a human-in-the-loop robot grasping system developed for assistive robots. This system contains a novel object movement-based grasp quality measure, a vision module, a deep learning-based grasp detection module, a grasp refinement module, and a user-friendly control interface. The grasp quality measure determines grasp poses' quality by analyzing the interactions between the gripper and the object through their projections in the image space. The real robot grasping results show that our grasp quality measure is practical and intuitive. Moreover, our grasp quality measure outperformed the other three well-known quality measures in generating grasp poses that cause minimum object movements in the method comparison experiment. The system's vision module can perform (2D and 3D) visual perception of the object and scene and recognize and segment objects in the scene. Multi-view point cloud integration and Point Pair Features (PPF) surface matching were used in the 3D perception process. A state-of-the-art object recognition and segmentation neural network (Facebook Detectron2) was used for object recognition and segmentation in 2D. The grasp detection module combined a state-of-the-art grasp detection neural network (Contact GraspNet) with our grasp pose filtering technique to form a module that can provide initial grasp pose options in different grasp directions. The grasp refinement module used our novel grasp pose refinement method for finetuning the initial grasp pose generated by the grasp detection module. The grasp pose refinement includes three stages. The first stage is the grasp direction (θ_x, θ_y) tuning for better gripper alignment to the object's approach and contact surfaces. The second stage is the grasp depth (z) tuning to ensure stable grasping. The third stage is the pose projection parameters

(x, y, θ_z) tuning for minimizing the object movements in grasping. A user-friendly graphical interface was created to provide the user feedback on the status of the grasping system, which allows the user to supervise and correct the system when errors occur. In scenarios where the high autonomy grasping mode does not work, the user can teleoperate the robot for grasping with an assisted velocity control method. To test the performance of the developed HitL robot grasping system, a robot grasping experiment was conducted to compare our grasping system with a state-of-the-art baseline grasping system. The experimental results proved that our grasping system, compared to the baseline system, can generate more accurate grasp poses, which had higher grasp success rates and caused less object movement during grasping.

While our robot grasping system was developed to be reliable, accurate, and easy to use, there are three main limitations. The first is that the grasping system is specialized for parallel-jaw grippers. We focused on pushing the limit of the parallel-jaw gripper grasping techniques rather than tackling the general robot grasping problem, as most of the current assistive robots use these grippers. The second is that our grasping system requires more time for performing object 3D perception and finetuning of the initial grasp pose. This is a trade-off between the speed and accuracy of the robot grasping. The extra time required by our grasping system is composed of three parts, the object silhouette extraction time (0.1 ~ 1.4 seconds, depending on the size of the object), the grasp refinement time (0.2 ~ 0.4 seconds), and the time for the robot moving to obtain an additional view of the object (which depends on the robot's speed and trajectory length). The third limitation is that the high autonomy grasping mode depends on the accuracy of the visual perception of the object, which is a common problem for all computer vision-guided robot grasping.

In future works, there are three aspects of the current system can be improved:

1. We will improve the vision module's 3D perception accuracy by updating the camera hardware and developing a more intelligent view pose selection method to find optimal view poses for perceiving the object and scene.
2. We will add task-related pose selection criteria in the grasp pose filtering technique, which enables task-oriented grasping.
3. In this work, we found effective features for grasp evaluation and refinement. Our next step is using deep learning to extract those features to optimize our grasp evaluation and refinement methods. For the grasp evaluation, we can train a convolutional neural network with data that contain object movement predictions in the grasp pose labels to obtain a network that can predict the object translation, rotation, and slippage. The final grasp quality score prediction can be calculated by combining the network-extracted object movement feature scores. Similarly, we can construct a grasp pose refinement network trained with data that reflect the features we used to formulate the refinement process presented in Chapter 6. Deep learning can significantly increase feature extraction efficiency for our grasp quality measure and grasp pose refinement method.

References

[1] "World Health Organization," [Online]. Available: https://www.who.int/.

[2] WHO, "World Report on Disability," World Health Organization, Geneva, 2011.

[3] S. W. Brose, D. J. Weber, B. A. Salatin, G. G. Grindle, H. Wang, J. J. Vazquez and R. A. Cooper, "The Role of Assistive Robotics in The Lives of Persons with Disability," *American Journal of Physical Medicine & Rehabilitation*, vol. 89, no. 6, pp. 509-521, 2010.

[4] A. S. Prabuwono, K. H. S. Allehaibi and K. Kurnianingsih, "Assistive Robotic Technology: a Review," *Computer Engineering and Applications Journal*, vol. 6, no. 2, pp. 71-78, 2017.

[5] M. Kyrarini, F. Lygerakis, A. Rajavenkatanarayanan, C. Sevastopoulos, H. R. Nambiappan, K. K. Chaitanya, A. R. Babu, J. Mathew and F. Makedon, "A Survey of Robots in Healthcare," *Technologies*, vol. 9, no. 1, p. 8, 2021.

[6] I. A. Șucan and S. Chitta, "MoveIt," [Online]. Available: moveit.ros.org.

[7] T. Tan, R. Alqasemi, R. Dubey and S. Sarkar, "Formulation and Validation of an Intuitive Quality Measure for Antipodal Grasp Pose Evaluation," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6907-6914, 2021.

[8] A. Bicchi and V. Kumar, "Robotic Grasping and Contact: A Review," in *IEEE International Conference on Robotics and Automation*, San Francisco, 2000.

[9] H. Zhang, J. Tang, S. Sun and X. Lan, *Robotic Grasping from Classical to Modern: A Survey*, arXiv, 2022.

[10] J. Bohg, A. Morales, T. Asfour and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289-309, 2014.

[11] H. Tian, K. Song, S. Li, S. Ma, J. Xu and Y. Yan, "Data-driven robotic visual grasping detection for unknown objects: A problem-oriented review," *Expert Systems with Applications*, vol. 211, 2023.

[12] I. Lenz, H. Lee and A. Saxena, "Deep Learning for Detecting Robotic Grasps," in *Robotics: Science and Systems IX*, Berlin, Germany, 2013.

[13] K. Kleeberger, R. Bormann, W. Kraus and M. F. Huber, "A Survey on Learning-Based Robotic Grasping," *Current Robotics Reports,* vol. 1, pp. 239-249, 2020.

[14] Z. Yin and Y. Li, "Overview of robotic grasp detection from 2D to 3D," *Cognitive Robotics,* vol. 2, pp. 73-82, 2022.

[15] R. Newbury, M. Gu, L. Chumbley and e. al., *Deep Learning Approaches to Grasp Synthesis: A Review,* arXiv preprint arXiv:2207.02556v2, 2022.

[16] S. Ainetter and F. Fraundorfer, "End-to-end Trainable Deep Neural Network for Robotic Grasp Detection and Semantic Segmentation from RGB," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, 2021.

[17] I. Lenz, H. Lee and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research,* vol. 34, no. 4-5, pp. 705-724, 2015.

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

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

[20] V. Satish, J. Mahler and K. Goldberg, "On-Policy Dataset Synthesis for Learning Robot Grasping Policies Using Fully Convolutional Deep Networks," *IEEE Robotics and Automation Letters,* vol. 4, no. 2, pp. 1357 - 1364, 2019.

[21] M. Vohra, R. Prakash and L. Behera, "Real-time Grasp Pose Estimation for Novel Objects in Densely Cluttered Environment," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2019.

[22] H. Kasaei and M. Kasaei, "MVGrasp: Real-time multi-view 3D object grasping in highly cluttered environments," *Robotics and Autonomous Systems,* vol. 160, 2023.

[23] M. Gualtieri, A. Ten Pas, K. Saenko and R. Platt, "High precision grasp pose detection in dense clutter," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, 2016.

[24] M. Breyer, J. J. Chung, L. Ott and e. al., "Volumetric grasping network: Real-time 6 dof grasp detection in clutter," in *Conference on Robot Learning*, 2021.

[25] M. Sundermeyer, A. Mousavian, R. Triebel and D. Fox, "Contact-GraspNet: Efficient 6-DoF Grasp Generation in Cluttered Scenes," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[26] W. Wei, Y. Luo, F. Li and e. al., "GPR: Grasp Pose Refinement Network for Cluttered Scenes," in *IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, 2021.

[27] Z. Zhao, H. Yu, H. Wu and X. Zhang, "6-DoF Robotic Grasping with Transformer," *arXiv preprint arXiv:2301.12476*, 2023.

[28] H. Fu, X. Mei, Z. Zhang and e. al., "Point Pair Feature based 6D pose estimation for robotic grasping," in *IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Chongqing, China, 2020.

[29] H. Wen, J. Yan, W. Peng and Y. Sun, "TransGrasp: Grasp Pose Estimation of a Category of Objects by Transferring Grasps from Only One Labeled Instance," in *Computer Vision--ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23--27, 2022, Proceedings, Part XXXIX*, 2022.

[30] "Repository for OpenCV's extra modules," [Online]. Available: https://github.com/opencv/opencv_contrib.

[31] B. Drost, M. Ulrich, N. Navab and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, USA, 2010.

[32] G. Marullo, L. Tanzi, P. Piazzolla and E. Vezzetti, "6D object position estimation from 2D images: a literature review," *Multimedia Tools and Applications*, pp. 1-39, 2022.

[33] Y. Zhu, M. Li, W. Yao and C. Chen, "A Review of 6D Object Pose Estimation," in *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2022.

[34] P. Ni, W. Zhang, X. Zhu and Q. Cao, "Learning an end-to-end spatial grasp generation and refinement algorithm from simulation," *Machine Vision and Applications*, vol. 32, no. 10, 2021.

[35] J. Cai, J. Cen, H. Wang and M. Y. Wang, "Real-Time Collision-Free Grasp Pose Detection With Geometry-Aware Refinement Using High-Resolution Volume," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, 2022.

[36] C. R. Qi, L. Yi, H. Su and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, CA, USA, 2017.

[37] H. Lei, N. Akhtar and A. Mian, "Spherical Kernel for Efficient Graph Convolution on 3D Point Clouds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3664 - 3680, 2021.

[38] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo and R. Girshick, "Detectron2," 2019. [Online]. Available: https://github.com/facebookresearch/detectron2.

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

[40] C. Ferrari and J. Canny, "Planning optimal grasps," in *IEEE International Conference on Robotics and Automation*, Nice, France, 1992.

[41] A. Miller and P. Allen, "Graspit! A versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110-122, 2004.

[42] M. Roa and R. Suárez, "Grasp quality measures: review and performance," *Autonomous Robots*, vol. 38, pp. 65-88, 2015.

[43] I.-M. Chen and J. Burdick, "Finding antipodal point grasps on irregularly shaped objects," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 4, pp. 507-512, 1993.

[44] A. Morales, G. Recatala, P. Sanz and A. d. Pobil, "Heuristic vision-based computation of planar antipodal grasps on unknown objects," in *IEEE International Conference on Robotics and Automation*, Seoul, South Korea, 2001.

[45] C. Davidson and A. Blake, "Error-tolerant visual planning of planar grasp," in *Sixth International Conference on Computer Vision*, Bombay, India, 1998.

[46] R. Krug, A. J. Lilienthal, D. Kragic and Y. Bekiroglu, "Analytic grasp success prediction with tactile feedback," in *IEEE International Conference on Robotics and Automation*, Stockholm, Sweden, 2016.

[47] K. Hsiao, S. Chitta, M. Ciocarlie and E. G. Jones, "Contact-reactive grasping of objects with partial shape information," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, 2010.

[48] H. Jeong and J. Cheong, "Evaluation of 3D Grasps with Physical Interpretations Using Object Wrench Space," *Robotica*, vol. 30, no. 3, p. 405–417, 2012.

[49] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy and D. Cournapeau, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods,* vol. 17, pp. 261-272, 2020.

[50] "ROBOTIQ," [Online]. Available: https://robotiq.com/products/2f85-140-adaptive-robot-gripper.

[51] Q.-Y. Zhou, J. Park and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," January 2018.

[52] Y. Chen, "A tutorial on kernel density estimation and recent advances," *Biostatistics & Epidemiology,* vol. 1, no. 1, pp. 161-187, 2017.

[53] F. Pedregosa, G. Varoquaux, A. Gramfort and a. et, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[54] R. Szeliski, in *Computer Vision: Algorithms and Applications 2nd Edition*, 2021, pp. 51-60.

[55] S. Maneewongvatana and D. M. Mount, "On the efficiency of nearest neighbor searching with data clustered in lower dimensions," in *International Conference on Computational Science*, 2001.

[56] T.-C. Lee, R. L. Kashyap and C.-N. Chu, "Building skeleton models via 3-D medial surface axis thinning algorithms," *CVGIP: Graphical Models and Image Processing,* vol. 56, pp. 462-478, 1994.

[57] S. Van der Walt, S. J. L, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart and T. Yu, "scikit-image: image processing in Python," *PeerJ,* vol. 2, p. e453, 2014.

[58] "Robotic Operating System," [Online]. Available: https://www.ros.org.

[59] "Baxter Robot," [Online]. Available: https://robots.ieee.org/robots/baxter/.

[60] "KINOVA," KINOVA, [Online]. Available: https://www.kinovarobotics.com/product/gen3-robots.

[61] "Intel RealSense," [Online]. Available: https://www.intelrealsense.com/depth-camera-d435i/.

[62] C. Xie, Y. Xiang, A. Mousavian and D. Fox, "Unseen Object Instance Segmentation for Robotic Environments," *IEEE Transactions on Robotics,* vol. 37, no. 5, pp. 1343 - 1359, 2021.

[63] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools,* 2000.

[64] T. Zinsser, J. Schmidt and H. Niemann, "A refined ICP algorithm for robust 3-D correspondence estimation," in *International Conference on Image Processing*, Barcelona, Spain, 2003.

[65] C. R. Harris, K. J. Millman, S. J. v. d. Walt, R. Gommers, P. Virtanen and D. Cournapeau, "Array Programming with NumPy," *Nature,* vol. 585, pp. 357-362, 2020.

[66] M. Qasim and O. Y. Ismael, "Shared Control of a Robot Arm Using BCI and Computer Vision," *J Eur Syst Automat,* vol. 55, pp. 139-146, 2022.

[67] R. Wen, K. Yuan, Q. Wang, S. Heng and Z. Li, "Force-Guided High-Precision Grasping Control of Fragile and Deformable Objects Using sEMG-Based Force Prediction," *IEEE Robotics and Automation Letters,* vol. 5, pp. 2762-2769, 2020.

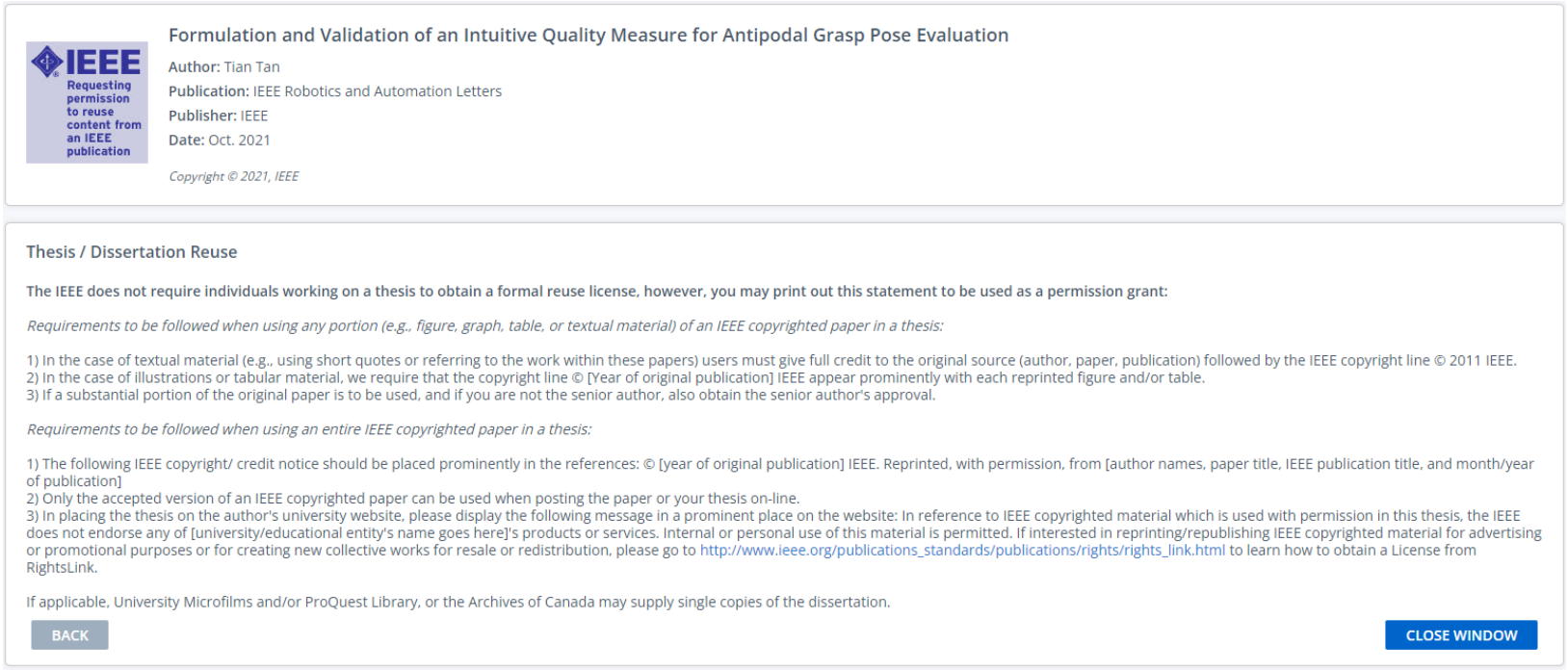
[68] J. Bohg and D. Kragic, "Learning grasping points with shape context," *Robotics and Autonomous Systems,* vol. 58, no. 4, pp. 362-377, 2010.

[69] S. Yu, D. Zhai, Y. Xia, H. Wu and e. al., "SE-ResUNet: A novel robotic grasp detection method," *IEEE Robotics and Automation Letters,* vol. 7, pp. 5238-5245, 2022.

[70] Y. Lu, B. Deng, Z. Wang and e. al., "Hybrid Physical Metric For 6-DoF Grasp Pose Detection," in *International Conference on Robotics and Automation (ICRA)*, Philadelphia, PA, USA, 2022.

Appendix A: Copyright Permissions

The contents in chapter 3 are from my paper "Formulation and Validation of an Intuitive Quality Measure for Antipodal Grasp Pose Evaluation" published in 2021 IEEE RA-L, the IEEE does not require individuals working on a thesis/dissertation to obtain a formal reuse license.



IEEE
Requesting permission to reuse content from an IEEE publication

Formulation and Validation of an Intuitive Quality Measure for Antipodal Grasp Pose Evaluation

Author: Tian Tan
Publication: IEEE Robotics and Automation Letters
Publisher: IEEE
Date: Oct. 2021

Copyright © 2021, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW