Single-Shot RGB-D Grasping of Objects Using a Multi-finger Robot: A Grasp Rectangle Approach with Post-Processing

by

Pouya Samandi

B.Sc. (Mechanical Engineering), IUST, 2019

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Applied Science

in the
School of Engineering Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2024

Declaration of Committee

Name:	**Pouya Samandi**
Degree:	**Master of Applied Science**
Title:	**Single-Shot RGB-D Grasping of Objects Using a Multi-finger Robot: A Grasp Rectangle Approach with Post-Processing**
Committee:	**Chair: Jie Liang** Professor, Engineering Science
	Kamal Gupta Co-Supervisor Professor, Engineering Science
	Mehran Mehrandezh Co-Supervisor Professor, Engineering and Applied Science University of Regina
	Mike Hegedus Committee Member Lecturer, Engineering Science
	Mehrdad Moallem Examiner Professor, Mechatronic Systems Engineering

Abstract

Grasping objects with robots is a complex challenge in the field of robotics. This research introduces a fast and dependable method for picking up objects of varying shapes and colors. The primary aim is to develop a flexible approach capable of handling a wide array of different objects. The proposed method is designed to function when objects are placed on a table, and the robot is positioned either above or in front of them.

Building upon an existing technique called the grasp rectangle, we employ a trained network to enhance the way we grasp objects. What sets this apart is our ability to expand the network's capabilities to work with robots equipped with multiple fingers. To achieve this, we incorporate a post-processing step into the network.

Our experiments validate the effectiveness of our approach. We achieve a successful object grasp rate of 94.4% when viewed from above and an accuracy of 95.6% when grasping from the side. These findings highlight the considerable potential of our method in addressing the challenging problem of robotic grasping, particularly in scenarios involving different object placements and colors.

Keywords: Robotic grasping; multi-finger robotic hand; deep learning; grasp rectangle method; object detection; object segmentation

Dedication

I dedicate this thesis to the unwavering pillars of my life:

To my beloved mother, Mahsa Fakhimi, whose boundless love, sacrifices, and unwavering support have been the guiding forces shaping my journey. Your strength and encouragement have been my constant inspiration.

To my father, Hossein Samandi, whose wisdom, resilience, and encouragement have been a source of motivation and a reminder that challenges are but stepping stones to success.

This work stands as a tribute to the enduring impact of family and mentors, whose influence has shaped not only my academic endeavors but also the person I have become.

Acknowledgements

Special thanks to Dr. Kamal Gupta and Dr. Mehran Mehrandezh for their invaluable guidance, mentorship, and scholarly insights that have illuminated my academic path. Your dedication to knowledge and commitment to excellence have left an indelible mark on my intellectual pursuits.

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

The future of manufacturing is undoubtedly tied to robotics, as these incredible machines have the ability to automate tasks that were once daunting or even impossible for humans. A captivating challenge in the world of robotics is to develop effective grasping techniques. Engineers have created various robotic hands, each with its unique shapes and Degree of freedoms [1]. Over time, researchers have explored different approaches, such as analytical methods [2], [3], geometric shape-based grasping [4], and machine learning-based methods [5]. However, many of these methods were limited to specific tasks or objects, lacking the flexibility required for handling diverse scenarios.

In recent years, the focus of research has shifted towards creating grasping systems with greater adaptability, enabling robots to handle a wide range of objects, regardless of their shapes and sizes. To achieve this, machine learning algorithms have emerged as invaluable tools for improving grasp accuracy and reducing processing time. By harnessing the potential of machine learning, researchers can train algorithms to elevate robotic grasping capabilities, making manufacturing processes more efficient and productive [6], [7], [8].

Nonetheless, it is essential to approach machine learning with caution, considering the wealth of knowledge developed before its prevalence. While machine learning algorithms have shown impressive results in controlled experiments, they may not always be entirely reliable, as their performance heavily relies on the quality of the dataset and training process [9].

1.1. Related works

1.1.1. Analytical grasping methods

Analytical grasping methods also known as geometry-based method, have been a prominent research focus in robotics for achieving reliable and efficient grasping of objects. These methods utilize shape matching and geometric reasoning to determine suitable hand configurations and contact points for successful grasps using mathematical models and analytical approaches to determine optimal hand configurations and contact points for successful grasps using mathematical models and analytical approaches.

Grasp Quality Metrics: One fundamental aspect of analytical grasping methods is the definition and computation of grasp quality metrics. Various metrics have been proposed, such as force-closure, form-closure, wrench-based, and friction-based metrics [2], [3]. In their work [10] presented a comprehensive survey of grasp quality metrics, providing insights into their mathematical foundations and practical applications.

In addition, the method that has been proposed by [11] uses triangular planes to analyze the contact points and corresponding normals. In [12], the authors tried to minimize the computational time for force closure using predefined constraints. Also, in [13] the authors decreased computational time by estimating the grasp wrench solely for contacts that align geometrically with parallel jaws.

Template-Based Approaches: Template-based approaches are one category of analytical methods. These methods utilize predefined hand configurations or templates that are designed or learned to match the shape of the object being grasped. For instance, [14] proposed a template-based grasping method where a library of grasp templates was created based on known object shapes. The template that best matched the object's shape was selected for grasp planning and execution.

Model-Based Approaches: Model-based approaches aim to construct a 3D model of the object and perform grasp planning based on this model [15] presented a model-based grasping system that utilized depth data from a range sensor to reconstruct the object's geometry and they tend to grasp object relied on the table. Grasp planning was then performed by analyzing the geometric properties of the reconstructed model, such as surface normal and curvature.

Additionally, spatial features have been extracted from household objects captured from a single viewpoint, facilitating their classification into various object primitives conducive to grasping [16]. These primitives offer a heuristic simplification of an object's shape, representing it with predefined geometric forms like cubes, cylinders, or spheres [17]. Objects may be represented by a single primitive or broken down into sub-primitives [18], [19].

Shape Matching and Feature Extraction: Shape matching is a fundamental component of analytical methods. Various techniques have been developed to extract features (Object’s surface, normals and e.g.) from object shapes and match them to

hand configurations. [4] is a shape context-based approach for grasp synthesis, where local shape descriptors were computed and matched to hand templates to determine suitable grasps.

Object Recognition and Pose Estimation: Accurate object recognition and pose estimation are crucial for an analytical method. [20] introduced a method that combined shape-based object recognition with pose estimation using RGB-D data. The system utilized a segmentation method to detect object and estimate the best grasp, to grasp the objects.

Limitations and Challenges: analytical methods face challenges in handling objects with complex shapes, occlusions, or pose variations. These methods rely on precise information about the object's geometry and friction coefficients. Also, they often rely on accurate object recognition and pose estimation, which can be challenging in cluttered environment. Furthermore, achieving real-time performance and adaptability to novel objects remain ongoing research challenges.

As a result, analytical methods offer a promising avenue for enhancing robotic grasping capabilities by leveraging the geometric properties of objects. Template-based and model-based approaches, along with shape matching and feature extraction techniques, contribute to enabling more robust and adaptive grasping. Incorporating learning-based approaches further enhances the grasp synthesis process. However, challenges such as handling complex object shapes, real-time implementation, and generalization to novel objects require further investigation. The combination of analytical methods with other grasping approaches, such as deep learning-based methods, holds potential for advancing robotic grasping in real-world scenarios.

1.1.2. Deep learning and geometric shape-based grasping methods

In the field of robotic grasping, researchers have explored various machine learning methods, often combining Shape-Based Techniques with Deep Learning models to improve the effectiveness of multi-fingered grasping [5], [21], [22]. Furthermore, certain learning-based methods leverage primitive shapes or similar objects to discern effective grasps [23], [24], [25].

One notable study, [26] introduces a deep learning architecture that predicts stable grasp positions for robotic hands using partial object views. The Convolutional Neural Network (CNN) is trained on RGBD image patches of known object models and estimates grasp quality metrics, such as force closure, without explicit calculations. During runtime, the system generates scene point clouds and meshes from a single viewpoint and utilizes the deep network to create heatmaps for potential fingertip and palm locations. By leveraging Simulated Annealing in a grasp simulator, the approach identifies stable grasps that align with visible object portions and correspond to low-energy locations on the heatmaps.

Another relevant study, [22] introduces the first active deep learning approach to robotic grasping, specifically addressing the challenge of data collection for multi-finger hands. The proposed method employs a multi-armed bandit formulation to select between three exploration strategies, allowing the grasp model to cover the configuration space of grasps and diverse objects more effectively.

limitations and challenges: One limitation is the requirement for substantial amounts of labeled data for training, making data collection and annotation challenging for real-world scenarios [9]. Moreover, ensuring safety and reliability is a significant challenge due to the vulnerability of deep learning models to training dataset and unpredictable behavior, which can lead to potential safety hazards and unexpected failures in robotic systems [9]. Additionally, the method described in the papers relies heavily on point clouds, and creating and maintaining accurate point cloud descriptions of objects can be a daunting task [6].

While the introduced methods show improved success rate and speed compared to traditional shape-based techniques [6], they still face challenges in terms of speed and complexity. Since these models heavily rely on the train dataset, using these models may not be easily applicable in industrial settings due to unpredicted insectaries. To address these issues, this thesis we propose a new method that is faster, with higher success rate, and significantly easier to use and train, offering potential advancements in robotic grasping for real-world applications.

1.1.3. Reinforcement Learning in robotic grasping

Reinforcement Learning (RL) has gained significant popularity in the field of robotic grasping due to its unique advantages over Deep Learning models and shape-based methods [27], [28], [29], [30], [31]. Unlike shape-based methods, it doesn't necessitate a complete model description of objects. Some prominent studies showcase the success of RL in robotic grasping:

[32] presents a scalable RL approach, QT-Opt, for learning dynamic manipulation skills in grasping tasks. Unlike static grasping methods, QT-Opt enables the robot to perform closed-loop vision-based control, continuously updating its grasp strategy based on real-time observations to optimize long-horizon grasp success. By utilizing over 580k real-world grasp attempts, the approach trains a deep neural network Q-function with over 1.2M parameters, achieving an impressive 96% grasp success rate on unseen objects.

[33] introduces a hierarchical RL framework for robotic grasping based on point clouds. The lower-level hierarchy learns multiple grasp types, while the upper-level hierarchy selects the appropriate grasp based on the object's point cloud. Through an autonomous collection of grasp datasets via trial and error, the approach iteratively improves grasping performance.

However, RL in robotic grasping does face some limitations and challenges. One major limitation is the high sample complexity, as RL algorithms often require a large number of interactions with the environment for effective learning. This process can be time-consuming and impractical for physical robots, raising concerns about wear and tear and safety issues. Additionally, collecting sufficient data for training RL models can be costly and may not fully capture the diverse range of real-world grasping scenarios. Furthermore, transferring learned policies from simulation to the real world poses a significant hurdle due to the reality gap between the two domains, potentially leading to suboptimal performance or failures in real-world deployments.

Considering these limitations and safety concerns, using RL for grasping objects may not always be the preferred strategy. Researchers need to address these challenges and explore other approaches to ensure the effectiveness and reliability of robotic grasping in practical applications.

1.1.4. Grasp Rectangle method

In this thesis, we are developing our method on top of a method called "grasp rectangle method". This method has been widely used in automation and robotics [30], [34], [35], [36]. This method involves creating a bounding box or rectangle around the optimal point of grasp, determined by five key parameters: the center of grasp (x and y coordinates), orientation (theta), width of finger opening (w), and length of finger opening (h) [37], [38], [39], [40]. These parameters (Figure 1.1) define the position, alignment, and gripping configuration required for effective and secure object manipulation [30]. By using these parameters, a rectangle representing the grasp can be created, which is why it is referred to as the "grasp rectangle method."

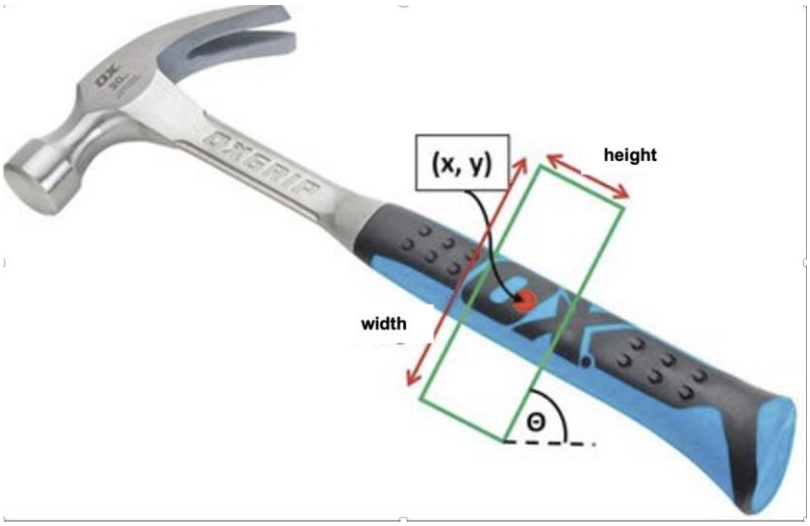


Figure 1.1: Grasp Rectangle method and outputs. Outputs are five parameters that define a rectangle in a plane [30].

This method was initially developed by [30], where they used a grasp rectangle to create bounding boxes around candidate grasp points. Since then, the method has been refined and adapted with different inputs and networks [34], [35], [36]. Some approaches use only RGB images as input [36], while others incorporate RGB-D information [34], [35]. In [34] employ hot encoding for network training, while others use object segmentation to create grasp rectangles [36]. Remarkably, some studies demonstrate the ability to generate grasp candidates in less than 50 milliseconds and create grasp rectangles for multiple objects from a single input.

Two widely used datasets, the Cornel [41] and Jacquard [42] grasp datasets, provide RGB-D images and corresponding grasp rectangles. The Cornel Dataset contains 240 objects with a total of 1,035 images and 8,019 grasps, while the Jacquard dataset comprises images from 11,000 objects, totaling 54,000 images and 1.1 million grasp rectangles. However, the top-view nature and solid backgrounds of these datasets can pose challenges when grasping objects with diverse or colorful backgrounds.

The method is well-suited for two-finger robots, grippers, and suction-based end-effectors due to the nature of the datasets used. However, it may struggle to create accurate grasp rectangles for objects with complex backgrounds or when viewed from different angles. Despite its promising features, to the best of our knowledge, no one has applied the grasp rectangle method to multi-finger robots for object grasping.

In this thesis, we present two main contributions. First, we extend the grasp rectangle method to be applicable for multi-finger robots. Second, we develop an algorithm that enables successful grasping of objects irrespective of their background or camera view.

Inspired by the promising results demonstrated in various studies and real-world experiments, we have chosen to leverage an existing network, specifically the one trained by [35], which achieved 94% accuracy on real experiment. Our objective is to build upon their work and adapt their method to accommodate a multi-finger robot for enhanced grasping capabilities.

1.1.5. Base method, GR-ConvNet

As mentioned earlier. We have developed our method, using grasp rectangle method. The are many studies suggesting a novel network architecture to increase the accuracy of this method. One of the best papers in this area is paper presented by [35].

In this project, a modular robotic system is presented to address the challenge of generating and executing effective grasps on unknown objects using antipodal robotic grasping. The approach involves a novel model called Generative Residual Convolutional Neural Network (GR-ConvNet) that can produce reliable antipodal grasps from multi-channel input in real-time. The performance of the proposed model is evaluated on established datasets and a diverse range of household objects.

The results demonstrate exceptional accuracy, achieving a state-of-the-art accuracy of 97.7% on the Cornell grasping datasets. Furthermore, a grasp success rate of 95.4% and 93% for household and adversarial objects is demonstrated, respectively, utilizing a 7-degree-of-freedom robotic arm.

Their network generates three images from which we can infer grasp rectangles for multiple objects. Figure 1.2 shows an overview of the proposed system architecture. The inference module acquires RGB and aligned depth images of the scene from the RGB-D camera. The images are pre-processed to match the input format of the GR-ConvNet. The network generates quality, angle, and width images, which are then used to infer antipodal grasp pose.

The outcomes produced by this neural network consist of five crucial parameters defining the grasp rectangle. These parameters, namely [x, y, theta, w, h] (Figure 1.1), have been comprehensively detailed in section 1.1.4 of the document. These values represent the horizontal position (x) and vertical position (y) of the grasp, the orientation angle (theta) of the rectangle, as well as its width (w) and height (h).

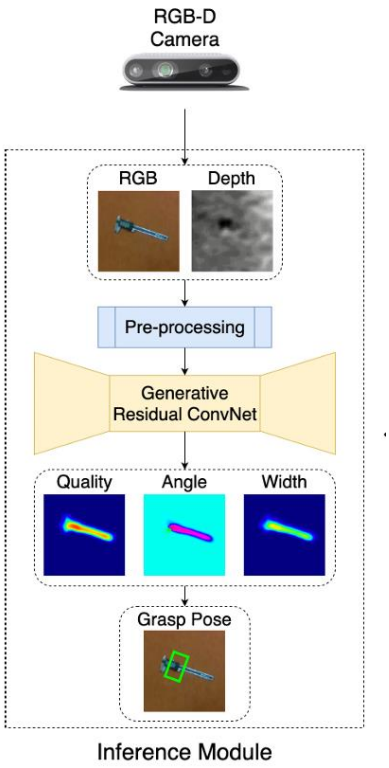


Figure 1.2: GR-ConvNet network architecture proposed in [36]. Input is an RGB-D image and output is the grasp rectangle parameters.

1.2. Contributions of this thesis

This project contains many contributions and they been explained below.

1. **Enhancing Grasping Techniques for Multi-Finger Robots:** While the Grasp Rectangle method has been widely utilized for object manipulation, its application with multi-finger robots remains unexplored territory until now. To the best of our knowledge, no prior work has leveraged this method's potential with multi-finger robotic systems.

2. **Efficient and Versatile Point Cloud-based Process:** In contrast to other prevalent methodologies that require the full point cloud definition of the object, our approach offers remarkable speed and versatility by utilizing partial point cloud representations. It effectively operates across a diverse array of objects, extending beyond the confines of pre-defined datasets. This adaptability significantly broadens its practical utility and impact.

3. **Expanding Grasp Rectangle into 3D Space:** While traditional Grasp Rectangle approaches have predominantly focused on top-view grasping scenarios, our method can grasp objects not only from a top view but also from the side of the object.

4. **Refinement and Search-Based Grasping:** Our project goes beyond mere utilization of network-generated rectangles. Through a tuning process, we not only enhance accuracy but also employ a search-based algorithm to determine the most viable grasping points. This dual-pronged approach ensures not just precision but also strategic grasp planning.

5. **Streamlined Implementation with Comparative Simplicity:** In contrast to methods requiring the combination of simulation and real-world images, our proposed technique provides relative simplicity and ease of implementation by using only a single-shot RGB-D image as input. This characteristic not only expedites integration into real-world applications but also fosters wider accessibility, ultimately driving practical adoption.

6. **Using Tactile Feedback for Asserting Enough Force on Objects:** In this project, we go beyond solely relying on vision to grasp objects. Instead, we utilize tactile

sensors installed on each link of the robotic hand to detect contact and apply adequate pressure or force for gripping objects. This approach significantly enhances our ability to grasp objects with precision and finesse.

The results of this thesis have been submitted to ICRA 2024 Conference: P. Samandi, K. Gupta, and M. Mehrandezh, "Enhancing Object Grasping Efficiency with Deep Learning and Post-Processing for Multi-Finger Robotic Hands," Manuscript submitted for presentation at the 2024 IEEE International Conference on Robotics and Automation (ICRA 2024).

The entire code for this project has been uploaded to the Google Drive folder containing all of our code and files with: https://drive.google.com/drive/folders/10gTKtssOeb2Z6OVQdz0Cdi3k93sMUfL9?usp=drive_link

Additionally, please refer to the video at this link for real experimental results demonstrating a multi-fingered robot grasping a variety of objects in real-time.

Chapter 2. Problem and Hardware Description

2.1. Problem description

The primary objective of this project is the development of a state-of-the-art algorithm, enabling a multi-finger robot to proficiently grasp a wide variety of objects with different shapes, sizes, and dimensions. Our aim is to achieve swift grasp calculations in real time or near real-time while upholding the highest levels of reliability and precision.

Our focus is on creating an algorithm that empowers the robot to adeptly grasp objects within its camera's field of view, with a particular emphasis on using only a single-shot image rather than scanning the object from multiple angles. This effort is concentrated on two main scenarios: objects observed from an overhead perspective, referred to as "viewing from top grasping," as shown in Figure 2.2, and objects viewed from the frontal angle, referred to as " viewing from side grasping," as shown in Figure 2.1.

The viewing from the top scenario often involves objects placed beneath the robot, a common occurrence in bin-picking operations. However, it's important to note that objects may not always be neatly arranged beneath the robot, and they can be positioned anywhere around it. Therefore, our algorithm is designed to handle grasping objects both beneath and in front of the robot to provide versatile and efficient object manipulation capabilities.

In this project, we are using tactile sensing to put enough pressure to grasp objects. In addition, we're focusing on only grasping objects once at a time. It means for each trial, there is only one object in the Camera view. For viewing from side grasping, we assume that the objects are sufficiently tall to be accommodated by our relatively large robotic hand.

Although our system has managed to grasp various objects including transparent objects, however, detecting transparent objects remains a challenge, and not all transparent objects are graspable using our current method. This challenge extends to white objects as well, and we will delve into this matter in further detail later. Consequently, our target objects primarily consist of items that feature colorful parts.

We assume that the dimensions of the objects are suitable for the hand and not excessively large or small. However, due to the limitations of the robotic hand, we exclude objects weighing more than 1kg from our grasp dataset. Additionally, we exclude deformable objects from our grasp dataset, as their unique nature of grasp and sensing presents distinct challenges. To summarize our objectives are:

- Develop an advanced algorithm for multi-finger robot grasping.
- Grasp various objects with speed, reliability, and precision.
- Use single-shot images for grasp calculations within the camera view.
- Handle viewing from the top and viewing from the side grasping scenarios.
- Adapt to objects placed around the robot, not just beneath it.
- Employ tactile sensing for precise pressure during grasping.
- Focus on single object grasping in each trial.
- The dimensions of the objects should be suitable for the hand to be grasped.
- Objects should have colorful parts to be recognized by our method.
- Objects should weigh less than 1kg.
- Objects should be solid and not deformable.
- Objects are placed on a flat surface, and they are not stacked on top of each other.

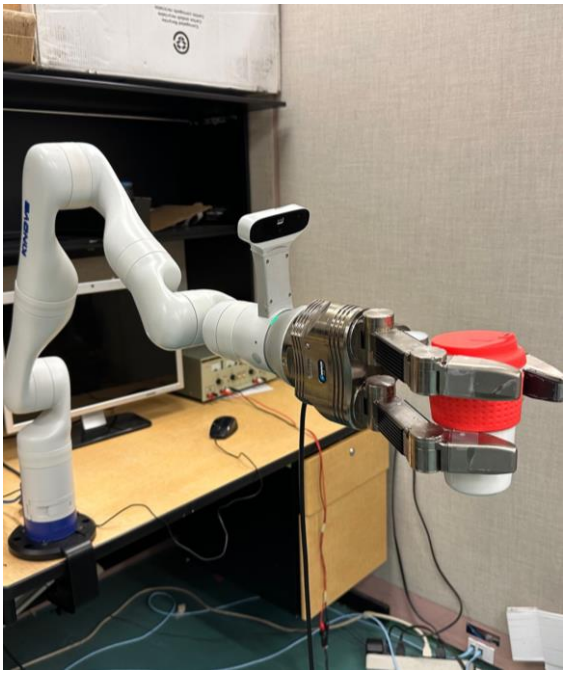


Figure 2.1: Schunk SDH robotic hand attached to the Kinova kortex Robot gen3 robotic arm. The robot is viewed from the side grasping configuration.

Figure 2.2: Schunk SDH robotic hand attached to the Kinova kortex Robot gen3 robotic arm. The robot is in the view from the top grasping configuration.

2.2. Objects to be grasped

Figure 2.3 and Figure 2.4 shows objects we tend to grasp viewing from the top and viewing from the side.

Figure 2.3: Objects we tend to grasp viewing from the side


Campbell's
Small and mighty.
Petit et puissant.
Google
5200mAh
HEINZ
KETCHUP
8m/26

Figure 2.4: Objects we tend to grasp viewing from the top

2.3. Setup explanation

2.3.1. Kinova Gen-3 robotic arm

The Kinova Gen3 [43] is a robotic arm platform developed by Kinova Robotics (Figure 2.5).

Key features of the Kinova Gen3 include:

1. This robot is equipped with a 1920 x 1080 (16:9) color module with 47 ± 3° field of view. Also, it's equipped with a Depth sensor with 480 x 270 (16:9) and a field of view of 72 ± 3°.
2. Safety features: The Gen3 is designed with safety in mind, with built-in safety mechanisms to prevent accidents and protect users and the surrounding environment.
3. ROS compatibility: The Gen3 is compatible with the Robot Operating System (ROS), which is a widely used framework for developing robotic applications. This

compatibility enables seamless integration with existing ROS-based systems and software libraries.

In addition, the camera attached to the arm is an Intel RealSense D415 depth camera with an accuracy of <2% at a 2 m distance [44]. It is developed by Intel Corporation, designed for capturing high-resolution depth images, and enables accurate depth perception in various applications. It features an infrared projector and dual infrared cameras, which work together to capture depth data with precision. The camera utilizes stereo vision technology to generate depth maps.

Figure 2.5: Kinova gen3 robotic arm with 7 degrees of freedom.

2.3.2. SCHUNK Dextrous hand 2.0 (SDH 2.0)

The SDH 2.0 (Schunk Dextrous Hand) is a highly versatile and advanced robotic hand developed by Schunk [45] (). Key Features of SDH are:

1. SDH has 3 articulated fingers, each with 2 DoF (Degree of Freedoms). In addition, it has a 7th DoF at the base of two of the fingers named as pivoting joint. This joint is

responsible for pivoting two base fingers as shown in Figure 2.6. The PIP (Proximal Interphalangeal) joint of a finger is the first joint from the fingertip, allowing bending and flexing of the finger. The DIP (Distal Interphalangeal) joint is the second joint at the tip of the finger, enabling further fine movement and control.

2. Sensors: The SDH is equipped with tactile sensors on the fingers, enabling it to perceive the forces and contact points during grasping. This sensor feedback provides valuable information for controlling the hand's grasp and ensuring safe interactions with the grasped object.

3. Control and Programming: The SDH can be controlled using various interfaces, such as ROS (Robot Operating System), which allows for seamless integration with other robotic systems and simplifies programming and operation. However, all the packages were compatible with ROS Kinetic on Ubuntu 16.04, is one of the challenges in this project was integration between the software as the rest of the code including Kinova packages was on Ubuntu 20.04 and ROS Noetic.

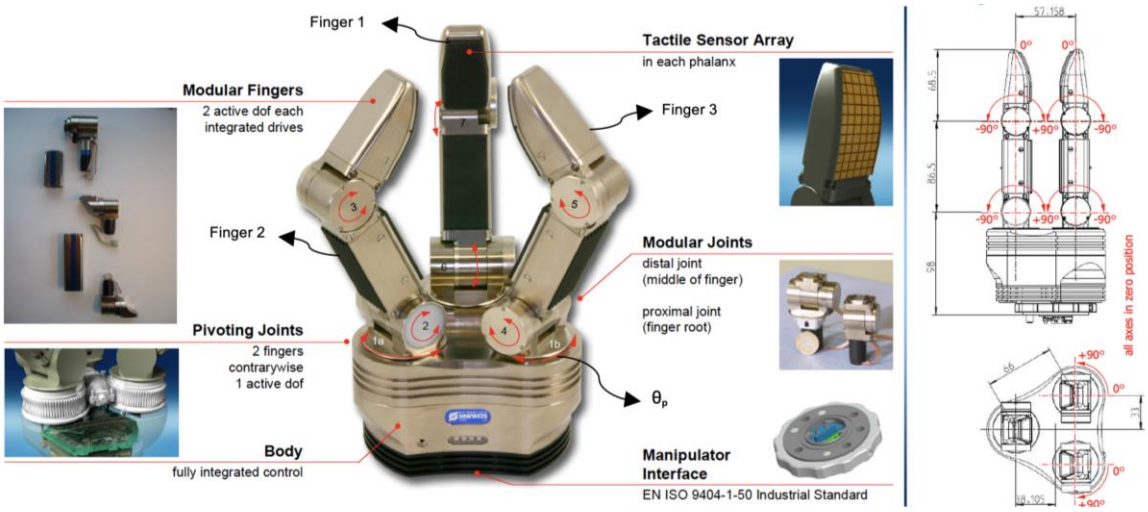


Figure 2.6: Schunk SDH robotic hand with 3-fingers [45]

Chapter 3. Grasp Strategy

In our method, we need five parameters to grasp an object which are: {P_e, θ_e, w, θ_p, G_t}. P_e [x, y, z] denotes the arm's end-effector positions, while θ_e [θ_x, θ_y, θ_z] signifies the orientation values. The 'w' parameter is the width of the grasp rectangle as shown in Figure 1.1. This parameter defines the initial finger separation width before grasping an object during the finger closure process. In other words, it shows how wide we should open fingers before grasping an object. The pivotal joint angle named θ_p (Shown in Figure 2.6) is the first joint of the robotic hand, responsible for pivoting the base of the finger 2 and finger 3). Lastly, 'G_t,' the grasp type parameter, specifies the desired grasp method: lateral, tripod, or power grasp (as illustrated in Figure 2.5). These parameters play a key role in solving inverse kinematics, allowing us to determine the robot's joint positions. For simplicity, we assume that the gripper's palm is perpendicular to the object, maintaining constant values for θ_x (yaw) and θ_y (pitch). This simplification enables us to define the end-effector's Cartesian pose as {P_e, θ_z}.

To determine these parameters, we employ the grasp rectangle method in conjunction with RGB-D images, as explained in section 1.1.5. While this method has exhibited over 90% accuracy in real-world experiments, it has primarily been designed for grippers with only two fingers. In Section 4.3, we have reported the results of using the grasp rectangle method with our multi-finger hand without any post-processing. Upon conducting these tests, it became apparent that although the grasp rectangle method performed admirably for two-finger robots, it encountered difficulties when applied to multi-finger robots. As a result**, we are proposing post-processing to use the grasp rectangle method for a multi-finger robot**. The Figure 3.1 shows the flowchart of the chapter 3.

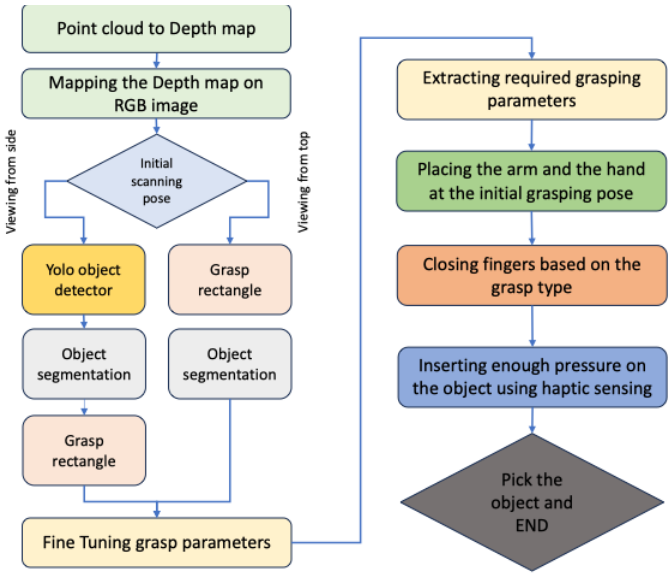


Figure 3.1: A flowchart of the chapter 3.

The camera, mounted on top of the robotic arm (Figure 2.2), captures information in the form of RGB images and point clouds from the environment. In Section 3.1, we convert these point clouds into a depth map. Subsequently, in Section 3.2, a calibration process synchronizes the depth information with RGB data, ensuring precise alignment between each pixel in the RGB image and the corresponding depth information.

The selection of the desired viewpoint for object grasping, whether from the side or top, guides our subsequent actions. When opting for viewing from the side, YOLOv5 is employed for object detection (Section 3.3), provides essential information. This data is then utilized for object segmentation, as detailed in Section 3.5, leading to the creation of a grasp rectangle (Section 3.4).

Contrarily, when choosing a top view for grasping, the grasp rectangle method is initially applied (Section 3.4). The rationale behind this approach is elaborated in Section 3.5. Subsequently, in the same section, we fine-tune the parameters extracted from the grasp rectangle method. The five crucial parameters - $\{P_e, \theta_e, w, \theta_p, G_t\}$ - are then derived from the available information.

Upon obtaining these vital parameters, the robot end-effector is positioned at the initial grasp pose (Section 3.4), and the robotic hand is set to an initial grasping configuration

based on the grasp type (Section 3.6). At the end, in Section 3.7, the fingers are closed, applying controlled pressure monitored by tactile sensing, culminating in the successful picking of the object.

The pseudo-code below shows step-by-step of the process of our method. The reason we differentiate the viewing from side and viewing from top (viewing from top) has been explained in section 3.5.

Algorithm 1

- *Placing the Robot at the Initial scanning pose and choosing the view*
- *Converting Point clouds to Depth map. (Section 3.1)*
- *Calibrating Depth and RGB image. (Section 3.2)*
- *If: viewing from top grasping*
  - *use the grasp rectangle method to detect the best grasp pose candidate (Section 3.4)*
  - *Implementing the Post-processing method and extracting required parameters. (Section 3.5)*
- *Else if: viewing from side grasping*
  - *Detecting the object in the environment with YOLOv5 (Section 3.3)*
  - *Using YOLO and Segmentation algorithm to create the mask of the object. (Section 3.5)*
  - *Implementing the Post-processing method and extracting required parameters. (Section 3.5*
- *End if*
- *Place the hand and fingers at the initial grasp configuration and close the fingers based on the grasp type (Section 3.6)*
- *Stop fingers after pressing enough pressure on the object using tactile sensing (section 3.7)*

3.1. Point clouds to depth image/map

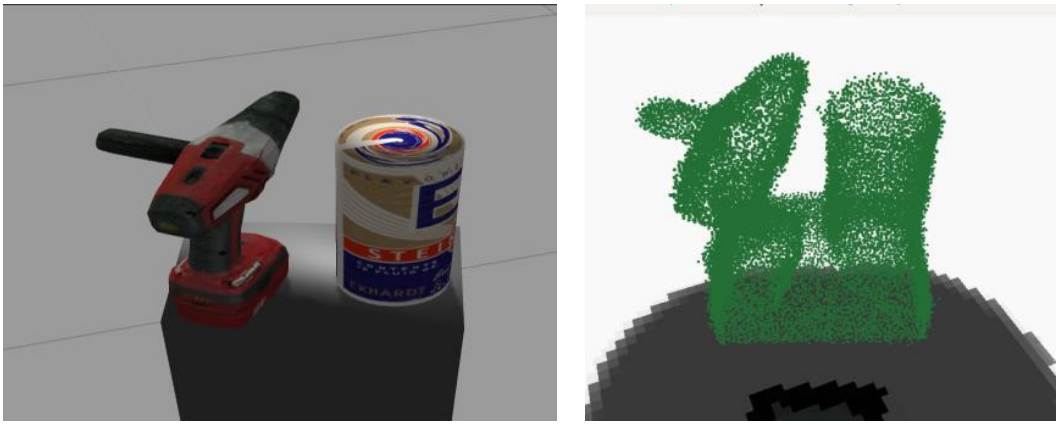
As mentioned in Section 2.3.1, our system incorporates a camera mounted on a robotic arm, accompanied by a lidar sensor that generates point clouds. The primary objective is to convert these point clouds into a Depth image as the input of our network is RGB-D. Each pixel represents the distance between the camera and the corresponding pixel in the 3D space. These point clouds encompass the spatial coordinates (X, Y, Z) of various physical shapes within the camera's field of view (Figure 3.2). We aim to assign each pose to an image, transforming the X and Y coordinates in 3D space into u and v coordinates

that correspond to specific pixels in an image, called depth image/map, with a defined resolution (640 x 480 image).

To generate a depth map from a point cloud, we leverage the Point Cloud Library (PCL). Specifically, we employ the voxel grid filter function to down-sample the cloud, with a grid size set to 0.01m. Additionally, we utilize the organized neighbor search to spatially organize points, with the search radius being 0.1m. Finally, the value for search parameter k, indicating the number of nearest neighbors considered during the organized neighbor search, is set be 10. These parameter values are integral to the process of transforming point clouds into depth images within our system.

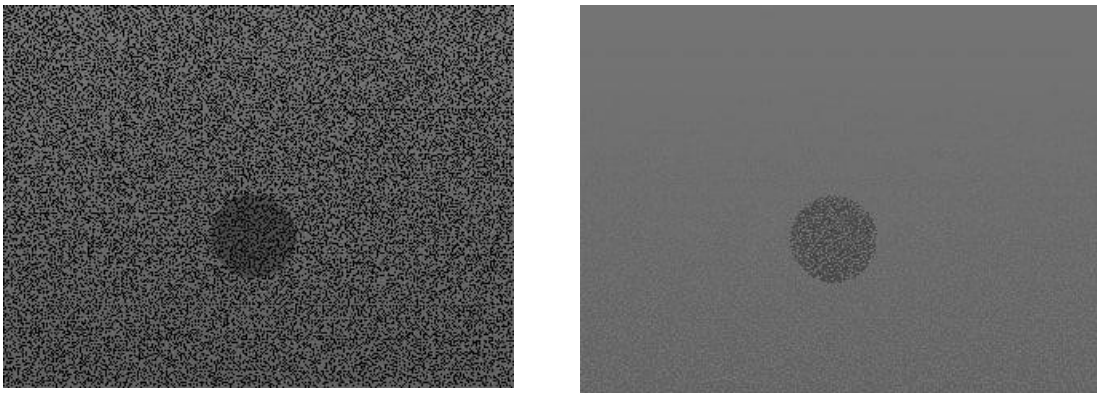
However, a challenge arises from the inherent noise present in the depth map obtained from the sensor. To address this issue, we have implemented a noise cancellation technique that involves uniform averaging of a specific portion of the image. By employing this approach, we replace the unassigned values in the Depth image, effectively reducing the noise. Figure 3.3 visually illustrates the impact of this noise cancellation process, offering a side-by-side comparison of the depth map before and after the application of the technique.

Through the implementation of this noise cancellation approach, our goal is to enhance the quality of the depth map. By reducing the noise, we can achieve more precise and reliable assignments of points to their respective positions in the designated empty image. This improvement plays a pivotal role in optimizing the overall performance of our system.



a) b)

Figure 3.2: a) a drill and a can in Simulation and b) point cloud presentation of objects [14]



a) b)

Figure 3.3: Depth map before and after noise cancelation filter of a can from the top view, a) noisy data b) denoised depth

3.2. Calibrating depth and RGB image

The Depth sensor captures a larger region compared to the RGB image, as depicted in the Figure 3.4 and as a result, the Depth and RGB images need to be aligned on each other and be calibrated. The calibration process involves a meticulous adjustment of the position and orientation of the images, specifically achieved by manipulating pixels in both the x and y directions.

The pixel movement process involves carefully adjusting the spatial positions of individual pixels in both the x and y directions. This adjustment is crucial to align corresponding features and structures in the larger-captured region of the Depth sensor with those in the RGB image. By precisely shifting pixels, we ensure that specific points in the scene captured by the Depth sensor align accurately with their counterparts in the RGB image, establishing a direct and faithful pixel-to-pixel correspondence.

Following this alignment, a trimming step is introduced to further refine the calibration. Trimming involves selectively removing or adjusting pixels at the periphery of the depth image, effectively eliminating any misalignments or inconsistencies that might have persisted after the initial pixel movement. Subsequently, to achieve uniformity in size, the trimmed depth image is resized to match the dimensions of the RGB image. This resizing process ensures that both images share the same scale. The result of this calibration is illustrated in the figure below, which showcases a side-by-side comparison of the images before and after the calibration process.

a) *b)* *c)*

Figure 3.4: calibrated RGB and Depth Image. a) shows the RGB image, b) corresponds to calibrated Depth image after trimming and shifting and c) shows the depth image without any calibration.

3.3. Object detection with YOLO

YOLO (You Only Look Once) [46] is an innovative object detection algorithm that has significantly advanced real-time object detection tasks. Unlike traditional detectors that require multiple passes over an image, YOLO performs detection in a single pass [47], [48], [49].

To accomplish this, YOLO divides an input image into a grid and predicts bounding boxes and class probabilities directly on this grid. It can identify multiple objects within the image, assigning corresponding class labels and confidence scores. YOLO operates at different scales to detect objects of varying sizes and aspect ratios.

One remarkable advantage of YOLO is its exceptional speed, enabling real-time object detection on high-resolution images and videos. This makes it highly suitable for applications requiring fast and accurate object detection, such as autonomous driving, video surveillance, and robotics.

YOLO has undergone several iterations, with each version improving accuracy and speed. Our project employs YOLOv5) [46], which has shown excellent performance in classification and detection tasks. However, it should be noted that YOLOv5 has a limited capacity to handle a wide range of objects.

Many objects we aim to grasp in our project are not present in its pre-trained dataset. To address this limitation, we fine-tuned YOLOv5 using a household dataset [50]. This dataset contains 388 annotated images with 1737 objects of the 20 classes. The training was conducted on a computer equipped with an Intel Core i9 CPU and an Nvidia GPU 3080 with 32 GB of memory. The dataset was split into training, evaluation, and test subsets as the portions 70%, 15%, and 15% respectively. The network was trained on the training dataset, and we utilized a 5-fold cross-validation method to evaluate its performance.

This network consists of three activation functions. a) **Localization Loss (Bounding Box Loss)**: To compute the difference between the predicted bounding box coordinates and the true bounding box we used Mean squared error (MSE). **b) Confidence Loss**: YOLO predicts a confidence score for each bounding box, indicating how confident the model is that the box contains an object and that the box is accurately localized. **c) Class Prediction Loss**: The class prediction loss is typically computed using categorical cross-entropy.

After training the network, we employed a ROS package provided by [51]. This package utilizes the trained model and operates solely on RGB images to generate bounding boxes around objects. Figure 3 provides an illustrative example of how YOLO functions.

By combining the power of YOLO with fine-tuning our specific dataset and integrating it into our ROS framework, we aim to achieve accurate and efficient object detection in real-world scenarios, facilitating successful grasping and manipulation tasks.

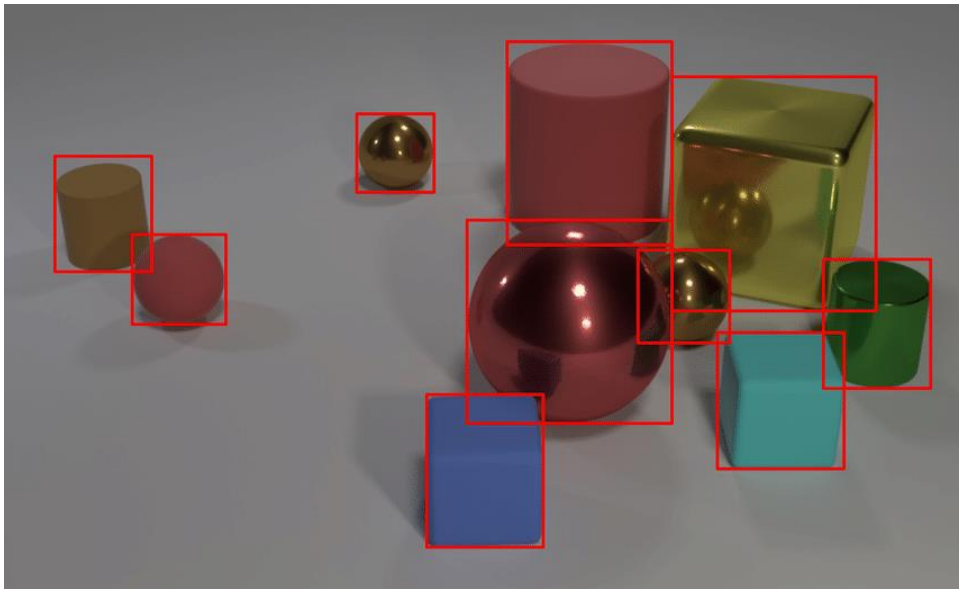


Figure 3.5: Using YOLO to detect objects

3.4. Grasp the rectangle and extract the spatial pose of the end-effector from it

As explained in Section 1.1.5, we use the GR-ConvNet network to generate grasp rectangles, yielding coordinates [x, y, θ, w] within a 2D camera frame (Figure 1.1). The process of extracting the pose of the end-effector from the information provided by grasp recatagle involves a progression from the image frame to the camera frame and ultimately to the base frame.

This transition involves the integration of the "z_c" value, representing the separation between the camera frame and the grasp center "g_c" of the image rectangle (Figure *3.16*). By employing defined mathematical expressions, the end-effector pose [X_c, Y_c, Z_c, θ_c] can be effectively computed within the context of the camera frame. These calculations enable the conversion of grasp rectangle information from the image frame to coordinates relevant for planning within the camera frame, facilitating subsequent robotic operations.

$$Xc = \frac{x - \frac{image\_width}{2}}{\frac{image\_width}{2}}.Zc \quad (1)$$

$$Yc = \frac{x - \frac{image\_height}{2}}{\frac{image\_height}{2}}.Zc \quad (2)$$

$$\theta_c = \theta \quad (3)$$

3.5. Tuning grasp rectangle method and extracting Grasp's pose parameters

As explained at the beginning of this chapter, our grasp planning approach necessitates the consideration of five parameters: $\{P_e, \theta_e, w, \theta_p, G_t\}$. While the values for P_e, θ_e, can be extracted from the output of the GR-ConvNet, combined with the Depth image and our predefined constraints, there remains the task of deducing the remaining parameters using the available data.

However, it is important to highlight that the initial output from the grasp rectangle network doesn't exactly match our specific needs. This is thoroughly discussed in Section 4.3, which emphasizes the crucial role of calibration. This calibration process fine-tunes certain parameters to achieve specific goals:

a) Ensuring that all fingers make contact with the object and none of them miss during the closing motion. b) Ensuring even pressure on the fingertips to prevent slipping or shifting of the object. c) Adjusting finger width for the best grip, avoiding both overly wide and too narrow grips that could affect the success of the grasp and the distribution of pressure. d) Deriving the angle for the pivoting joint and determining the grasp type based on available information, to configure the robotic hand appropriately for a successful grasp.

By carefully addressing these important considerations and refining parameter values, we improve the dependability and effectiveness of our multi-finger robotic grasping system. This enhancement empowers the robot to adeptly handle various objects in real-world situations.

In this project, we utilize the "GrabCut" function from the OpenCV library to perform object segmentation from the background and create a mask of the object. The mask is defined as the same region as the segmented object but with pixel values of 1 inside the object and 0 otherwise. GrabCut is an image segmentation method based on graph cuts. Starting with a user-specified bounding box around the object to be segmented, the algorithm estimates the color distribution of the target object and that of the background using a Gaussian mixture model. This technique requires an estimated bounding box around the object, which helps the algorithm accurately isolate the object. An intuitive approach would be to leverage the bounding boxes generated by an object detector like the YOLO [46]. However, relying solely on YOLO has its limitations, as it may not accurately detect the complete spectrum of objects. The original YOLO training dataset encompassed around 80 objects. We fine-tuned YOLOv5 using a household objects dataset [50] narrowing down its focus to 20 common household items. Impressively, after fewer than 50 training rounds, our model achieved an accuracy of 98.24\% on the evaluation dataset (Section 4.2). While this adaptation led to a commendable 97% success rate in detecting objects from a side-view grasping scenario (Section 4.2), its performance substantially deteriorated in the case of a top-view grasping scenario (Section 4.2), with the success rate dropping to below 60%. This less-than-optimal result, explored in (Section 4.2). can be attributed to YOLO's difficulty in precisely detecting smaller objects.

On the contrary, the grasp rectangle network excelled in identifying grasp rectangles across a broader array of objects, accommodating varying shapes and sizes. Nevertheless, this method demonstrated vulnerability to changes in background color and depth images. Instances featuring vivid backgrounds or sudden shifts in depth could undermine the effectiveness of this approach in generating feasible grasp rectangles.

When considering top-view grasping scenarios, the circumstances usually present a consistent, uniform background, and the depth map remains relatively stable as objects are positioned on a level surface. However, challenges arise in side-view grasping situations, where depth maps exhibit abrupt variations and display diverse, vibrant backgrounds. This discrepancy can be attributed to the training datasets in use, such as the Cornell [41] and Jaguar datasets [42], which primarily focused on objects placed on monochromatic surfaces at specific camera-to-object distances.

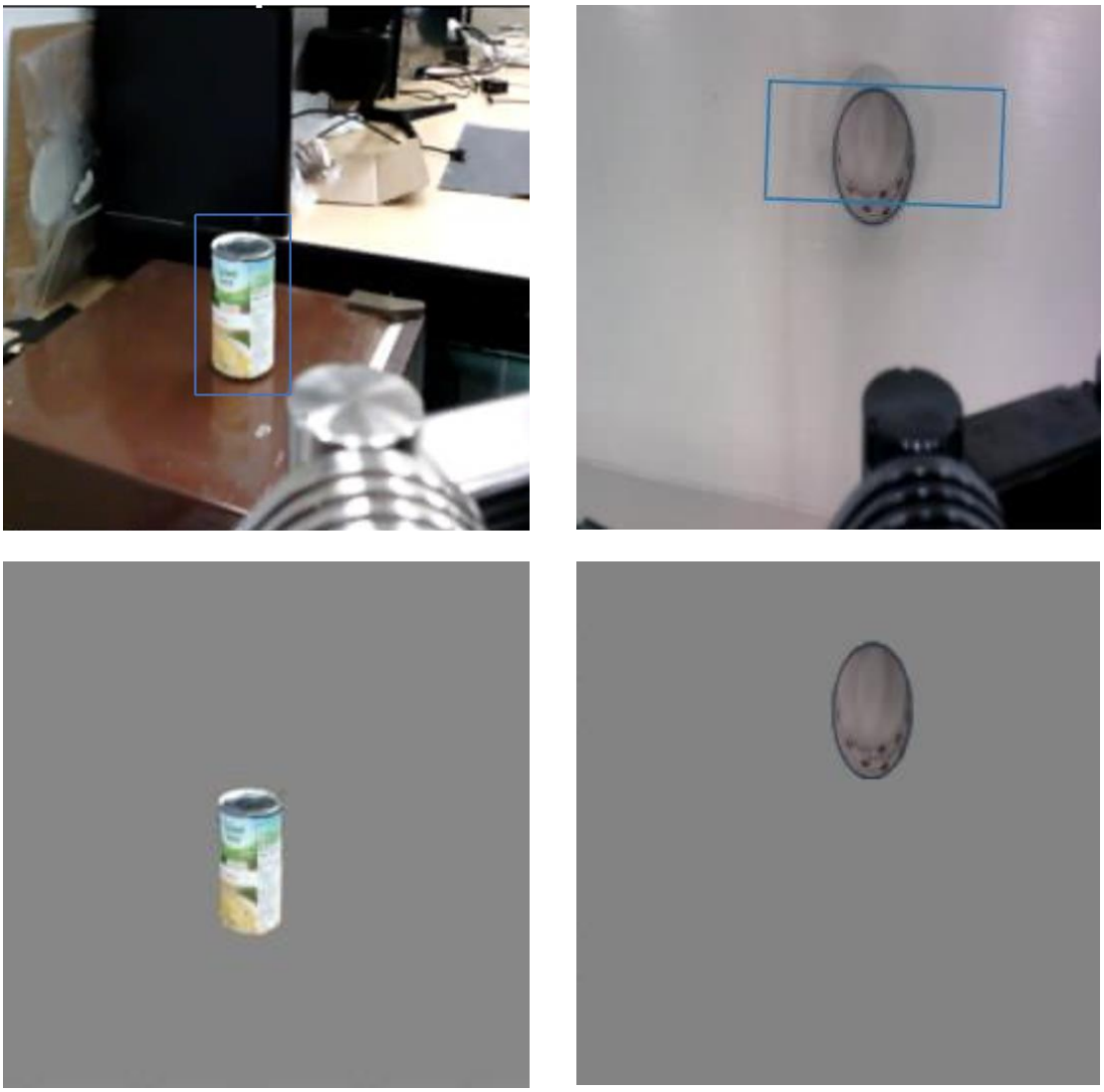


Figure 3.6: Top-left shows an object detected by YOLO. We use the bounding box and grabCut method to segment the object as shown in figure 12. c. Figure 12. b shows an object and grasp rectangle created by the GR-ConvNet network. We use this rectangle and grabCut method to segment the object from viewing from the top as shown in figure 12.d.

As a result, our approach capitalizes on the YOLO detector's ability to craft bounding boxes for the side-view grasping dataset, while the grasp rectangle network adeptly generates these bounding boxes for the top-view grasping context. Shows bounding boxes created by YOLO and GR-ConvNet. Figure 3.6 shows a bounding box created by YOLO from viewing from the side and a bounding box created by GR-ConvNet viewing from the top and their corresponding segmented object. The aluminum part in the downright corner of the Figures is one part of the robotic hand that is visible in the camera view. The accompanying Figure 3.7 illustrates the roadmap to determine all the necessary parameters for successfully grasping an object.

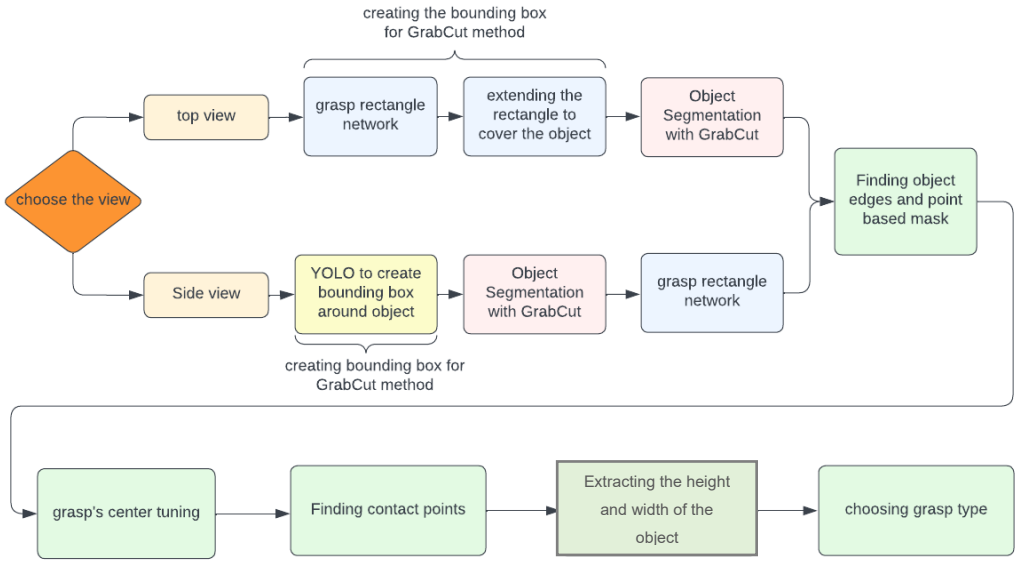


Figure 3.7: The Flowchart for our search-based processing (we call it SB-Convnet). It extracts the required grasp parameters for each view.

As depicted in Figure 3.7The object grasping process initiates by selecting the grasp view from the top or side. Grasp rectangles, outlined in Section 3.5 for viewing from the top or utilizing YOLO (Section 3.3), and define the bounding boxes. With this bounding box, we proceed to segment the object (Section 3.7). The green boxes in the figure symbolize a step-by-step post-processing method employed to refine the grasp center and extract the necessary parameters. The subsequent sections of this chapter elaborate on these steps in detail. In Section 3.5.1 we find object edges using the segmented object. In Section 3.5.2 we tune the grasp center points to have a more accurate grasp. In Section 3.5.3 and 3.5.4 we find the potential contact points and we choose the best candidate based on a search-based algorithm. In Section 3.5.5 and 3.5.6 we extract the height and width of the object from the RGB-D image. In section 3.5.7 we choose the grasp type using the Object's height and width.

3.5.1. Finding corresponding edges from the segmented object

After creating a segmented object, we used a computer vision technique with the object's mask to find its edges. We employed the Canny function from OpenCV, which is good at

detecting object edges. Then, we picked pixels from these detected edges to create a mask. This gave us a set of points along the object's edges for further analysis (Figure 3.8)

To figure out the edge normal at each point of the object boundary, we used the OpenCV Sobel method to calculate the gradient in the image coordinates. The normal vector at each point was determined by taking the cross-product of these gradient vectors. We then made sure the resulting vector had a length of 1, making it a unit normal.

To make the math easier, we rotated the mask using the theta value we got from another network. This theta represents the rotation angle of the rectangle that encloses the object.

This whole process helped us find the object's edges accurately and calculate normal angles at each pixel. These findings are crucial for our next steps, like planning how to manipulate and grasp the object.

a) *b)*

Figure 3.8: a) shows the segmented object and b) shows the edge of the object created by the canny method.

3.5.2. Grasp center tuning

As explained before, to achieve better results, we need to fine-tune the grasp center $g_c\{x_c,y_c\}$ in the image. To elaborate on this technique, we offer Figure 3.9 and Figure 3.10

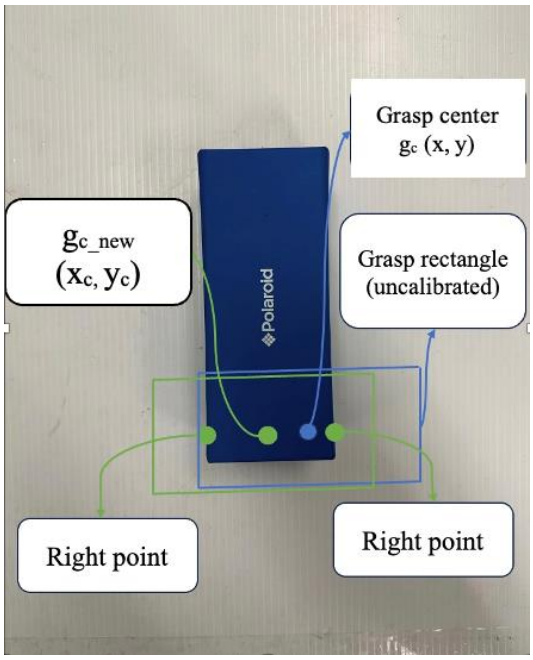


Figure 3.9: Calibrating grasp center along the x-axis

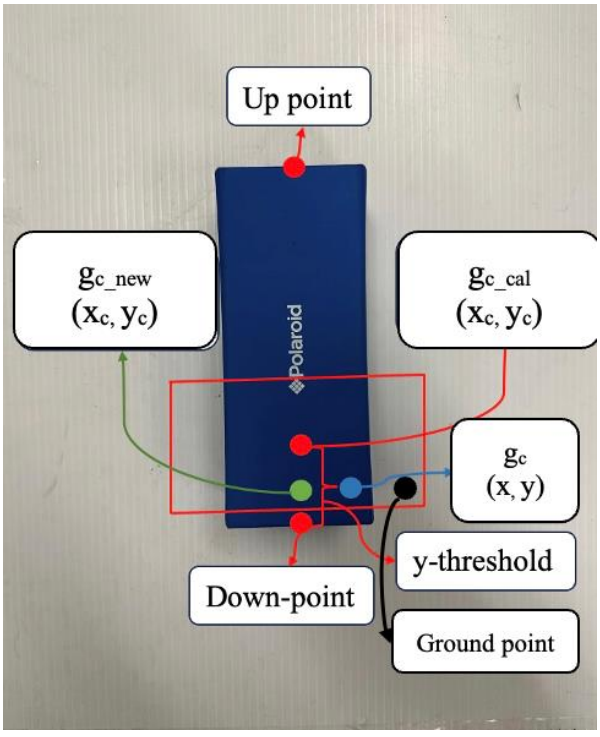


Figure 3.10: Calibrating grasp center along the y axis

as a visual guide, illustrating the sequential calibration process for grasp-center points. As shown in Figure 3.9, we transfer the grasp center (x) to the point in the middle of the right and left points and we name it $g_{c\text{-}new}$. The “right point” and “left point” sit at the object's extreme edges and share the same vertical position (y) as the “grasp center (GC)”.

Next, we focus on calibrating the y-coordinate. Our goal is to ensure there's enough space between the GC-new and the vertical edge of the object. To achieve this, we measure the distance between the $g_{c\text{-}new}$ and the "Up Point" and "Down Point". These points sit at the object's extreme edges and share the same horizontal position (x) as $g_{c\text{-}new}$. If these two distances happen to be smaller than a predefined threshold, we transfer the g_c.new in a direction that meets the threshold, and we call the calibrated point "$g_{c\text{-}cal}$". In addition, when the object is small, and the distance between the grasp center point and both the 'up point' and 'down point' falls below the threshold, we position the grasp center at the midpoint of these extreme points. This step guarantees that all the fingers have adequate space to potentially make contact with the object.

Importantly, this calibration process aims to optimize our grasp without significantly shifting the grasp rectangle from its initial position. This way, we can maintain the highest grasp quality based on the output of the original Network. The Pseudo code below shows a representation of the presented process.

Notes:

a) The mask-edge, a visual representation of the object's edge, adopts a structured format of [224, 224] pixels, where a pixel value of "1" signifies the presence of an object edge.

b) The mask-edge's segmentation serves as a basis for subsequent calculations, notably mask-edge[:, y_c] representing the collection of x-values corresponding to points sharing the common y coordinate within the mask-edge picture. Conversely, mask-edge[x_c, :] encapsulates the y values of points aligning with the identical x coordinate xc within the mask-edge depiction.

Algorithm 2

- *Get grasp center = [x_c, y_c]*
- *right-point = [max(mask-edge[:, y_c]), y_c], left point = [min(mask-edge[:, y_c]), y_c],*
- *$x_{c\text{-}new}$ = mean (right-point, left-point)*
- *up-point = [$x_{c\text{-}new}$, max(mask-edge[$x_{c\text{-}new}$, :])], down-point =[$x_{c\text{-}new}$, min(mask-edge[$x_{c\text{-}new}$, :])]. Figure (16)*
- $\Delta y_1 = |y_c - up\text{-}point|$ and $\Delta y_2 = |y_c - down\text{-}point|$

- *If:* $\Delta y_1 < Threshold(20pixels)$
  - $y_{c\text{-}new} = y_c - Threshold + \Delta y_1$ *(bringing down the center point)*
- *else-If:* $\Delta y_2 < Threshold(20pixels)$
  - $y_{c\text{-}new} = y_c + Threshold - \Delta y_2$ *(bringing up the center point)*

In addition, Figure 3.10 shows a point named ground point. This point is been defined by going 10 pixels in the horizontal direction from the “right point”. We need this point as a reference for a point that is not on the object but is close to the object on the ground. This point will be important to find the height of the object in 3D space as explained in Section 3.5.6.

3.5.3. Calculating the contact points on the object from the image

In our approach to identifying the optimal grasping approach for objects, we employ a search-based algorithm. This method involves evaluating various grasp candidates and selecting the most efficient one. The initial step is to estimate where the fingers would make contact when the palm is positioned over the grasp center and the fingers are closed. This estimation requires determining the position of the finger bases within the image plane, as depicted in the 3D visualization presented in Figure *3.11*.

However, it's important to note that the system perceives the object through RGB-D data which is a 2D image, as illustrated in Figure *3.12*. To bridge the gap, we define the projection of finger bases on the image plane. For clarity, we introduce the concept of a "projection rectangle," represented by the blue rectangle in both Figure 3.12 and Figure 3.13. This rectangle connects all the projected finger base points. As a result, with knowledge of the dimensions (width and height) of the "projection rectangle" and the center point of the rectangle on the image plane, we can estimate the projection of the finger bases onto the image plane.

The dimensions of the projection rectangle have a linear relationship with the distance from the camera frame to the grasp center. Simply put, as the hand moves further from the object, the dimensions of the projection rectangle decrease, and vice versa. To quantify this relationship, we introduce parameters "ku" and "kv," representing the width and height of the rectangle, respectively. Equations (5) and (6) express their computation:

$$k_u = k_{u0} - (Z_c / Z_0 - 1) \quad (5)$$

$$k_v = k_{v0} - (Z_c / Z_0 - 1) \quad (6)$$

Here, "Z_c" denotes the distance between the camera frame and the grasp center, extracted from a Depth image (Figure *3.16*). Constants "Z_0," " k_{u0}" and " k_{v0}" require calibration for each robot and camera system; in our setup, these constants were determined to be approximately 600 mm, 20, and 15, respectively.

With the grasp center, "ku," and "kv" determined, we can estimate the projection of fingertips on the grasp plane. By incorporating this algorithm and considering these factors, we achieve precise positioning of the finger bases. This, in turn, allows us to identify the optimal contact points for achieving the best possible grasp, as detailed in Section 3.5.4.

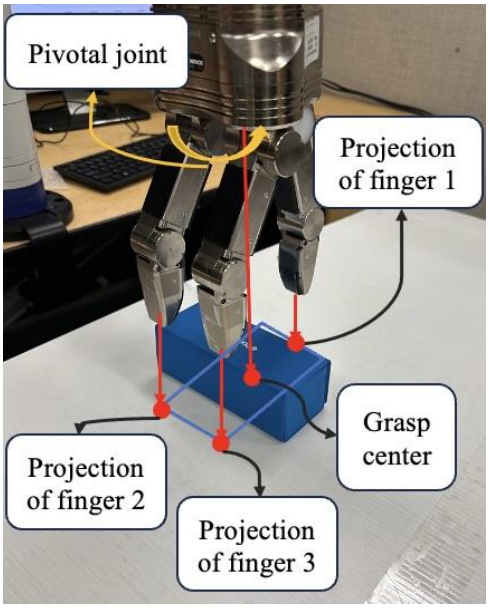


Figure 3.11: projection of fingertips and palm center (grasp-center) on the ground from the third-person view.

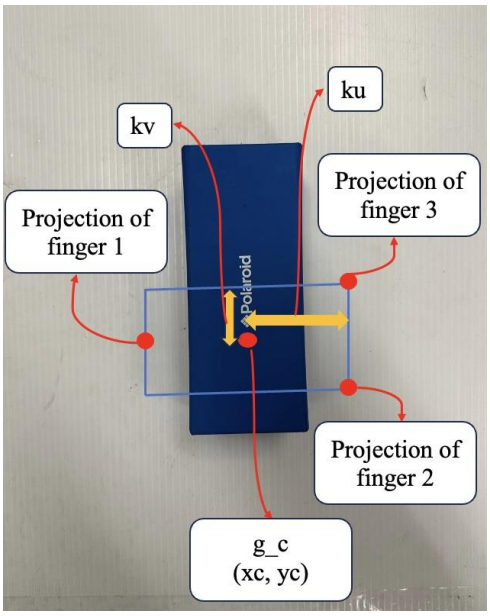


Figure 3.12: projection of fingertips and palm center (grasp-center) on the ground from the camera view.

3.5.4. Choosing the best contact points

By analyzing the finger bases, we can establish a line that indicates the direction in which the fingers close. This line intersects with the edge of the object, revealing the contact points for each finger. It is important to remember that the pivotal_joint (θ_p) can rotate Finger 2 and Finger 3 around their base frame (Figure 2.6), resulting in potentially different contact points based on their rotation. Figure *3.13* serves as an example to help a better understanding of the concept. For simplicity, we only use a pivotal joint angle between 0 to 90 degrees with 10-degree interval. In addition, Finger 1 can only be closed in one direction which is the horizontal direction.

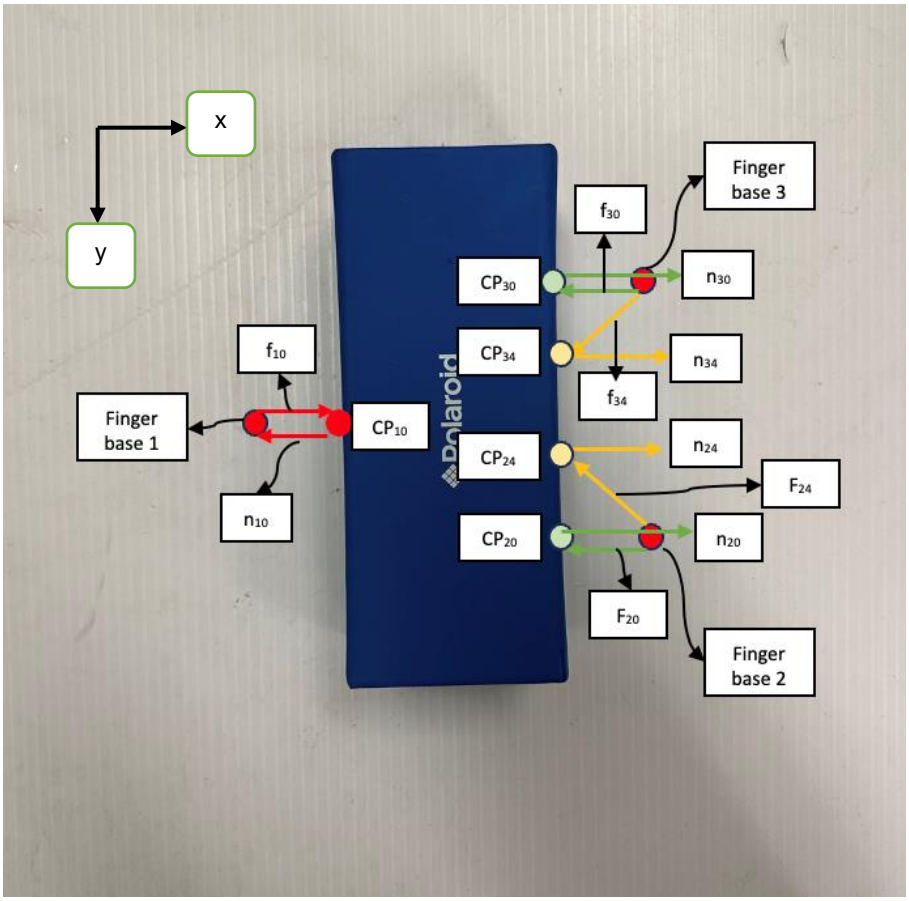


Figure 3.13: normal direction of contact points

Before explaining Figure *3.13* we need to define the parameters that have been presented in this figure.

$\hat{x}$: coordinate along the horizontal axis of image coordination.

$\hat{y}$: coordinate along the vertical axis of the image coordinate.

$\boldsymbol{\theta_{p,j}} \in \{0, 10, 20, \ldots, 90\}$ $and, j \in \{0, 1, \ldots, 9\}$, where $\theta_{p,j}$ is a pivotal joint angle between 0 to 90, (e.g: $\theta_{p,0} = 0$ and $\theta_{p,1} = 10$ and $\theta_{p,9} = 90$)

$\boldsymbol{f_{i,j}}$: direction of finger{i} ($i \in \{1, 2, 3\}$) and j_{th} closing direction of pivotal joint where: $j \in \{0, 1, \ldots, 9\}$. For finger 2, the $f_{2,j} = \theta_{p,j}$ and for finger 3, the $f_{3,j} = -\theta_{p,j}$.

$CP_{i,j}$: A contact point refers to the location where the closing finger $\{i\}$ makes contact with the object, applying sufficient pressure to halt its movement. Refer to section 3.7 for detailed insights into how fingers detect contact and exert adequate pressure on the object. These points are pixels coordinated in the image plane.

$\boldsymbol{n_{i,j}}$: normal direction of the surface at the contact point for finger $\{i\}$ and j_{th} closing direction of pivotal joint where $j \in \{0, 1, \ldots, 9\}$. The values are in degrees and correspond to the rotation concerning the horizontal line. The normal direction of the surface (edge) of the object has been extracted using the Sobel method explained in Section 3.5.1.

Our objective is to determine the pivotal joint angle ($\theta_{p,j}$) that minimizes the difference between finger directions and surface normals. These rotational disparities are quantified by Equation 7, where $\Delta\theta_j$ is the sum of absolute differences between $f_{i,j}$ and $n_{i,j}$ across all fingers. Equation 8 defines the rotational difference vector $\widehat{\Theta}$, and our goal is to identify the minimum value in this vector along with the corresponding $\theta_{p,j}$ value. For example, if $\Delta\boldsymbol{\theta_4}$ represents the minimum of $\widehat{\Theta}$, then: $\boldsymbol{j = 4}$ and $\boldsymbol{\theta_{p,j} = 40}$ and best contact points for each finger be at $CP_{i,4}$.

$$\Delta\theta_j = \sum_{i=1}^{3}(|f_{i,j} - n_{i,j}|) \qquad (7)$$

$$\widehat{\Theta} = [\Delta\theta_0, \Delta\theta_1, \ldots, \Delta\theta_9] \qquad (8)$$

For "finger 1," it's crucial to emphasize that this finger is restricted to movement along a single direction, as illustrated by the red arrow. Unlike other fingers, finger 1 lacks rotational flexibility; it can only close parallel to the horizontal line in the image. Then: $n_{1,j} = 0^o$ for $j \in \{0, 1, \ldots, 9\}$.

In contrast, "Finger 2" and "Finger 3" have more flexible motion. They can pivot relative to each other, as shown by green and yellow arrows. In this example provided in Figure *3.13*, the pivotal joint angles of the green arrows are 0^o degrees and the yellow arrows are 40^o degrees.

As it can be seen in, Figure 3.13, the “0” degree pivotal joint results in a smaller difference in norms ($\Delta\theta$) for all fingers, indicating better alignment with the object's surface. Conversely, the “40” degree pivotal joint leads to a larger difference in norms and is less suitable.

In essence, this method enables us to optimize finger rotation for improved alignment with the object's surface normal, determining the most effective finger configuration for a successful grasp.

In addition, Figure *3.14* shows an example of a contact point on the object and the direction in which we close the fingers. This direction has been chosen based on the method explained in this chapter.

Figure 3.14: Estimated contact points and the best finger directions that minimize the normal difference for all three fingers

3.5.5. Finding the grasp width

By having the best contact points, we aim to establish a reliable grasp rectangle. By utilizing the grasp rectangle, we can not only provide a visual representation but also define important parameters such as grasp width and finger opening. this information is essential in defining grasp type and grasp height parameters.

In Section 3.5.4, we find the best *contact points when we close the fingers. Figure 3.15* the contact points of finger 2 and finger 3, signifying the corner points on one side of the grasp rectangle. Meanwhile, the contact point of finger 1 denotes the contact location

along the middle edge on the opposite side of the rectangle. This information allows us to determine the length and width of the grasp rectangle, as visualized in the figure.

To enhance the precision and clarity of the grasp definition and representation, we have chosen to increase the size of the grasp rectangle. In this particular project, the dimensions of the rectangle have been enlarged by 10 pixels on each side. Note that this enlargement value may vary for different images with different resolutions.

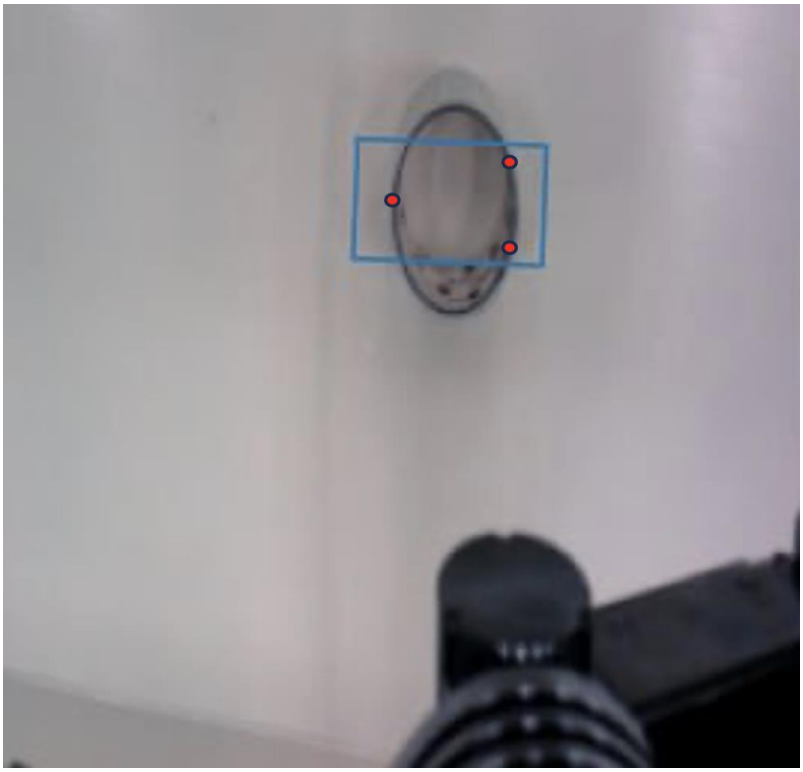


Figure 3.15: Extracting width from contact points

3.5.6. Finding the height of the object

To ascertain the height of an object, we rely on the measurement between the object's grasp center and a predetermined reference point located near the object. This reference point's precise distance is determined using a depth image. To

As depicted in Figure *3.16*, we leverage two key metrics, "Z_g" and "Z_c," to effectively characterize both the grasp type and the required depth for the grasp. "Z_c," signifies the distance between the camera and the center of the object's grasp. These metrics are derived from a depth image. On the other hand, "Z_g" represents the distance from the

camera to the ground or table surface, specifically denoting the distance to the predefined “ground point”. Figure 3.10shows the “ground point” on the image plane. As explained in Section 3.5.2, the ground point is defined by going 10 pixels in the horizontal direction from the “right point”.

Through the subtraction of "Z_g" from "Z_c," we can compute the height of the object. This information is pivotal in determining the appropriate grasp type and the optimal depth to which the fingers should descend.

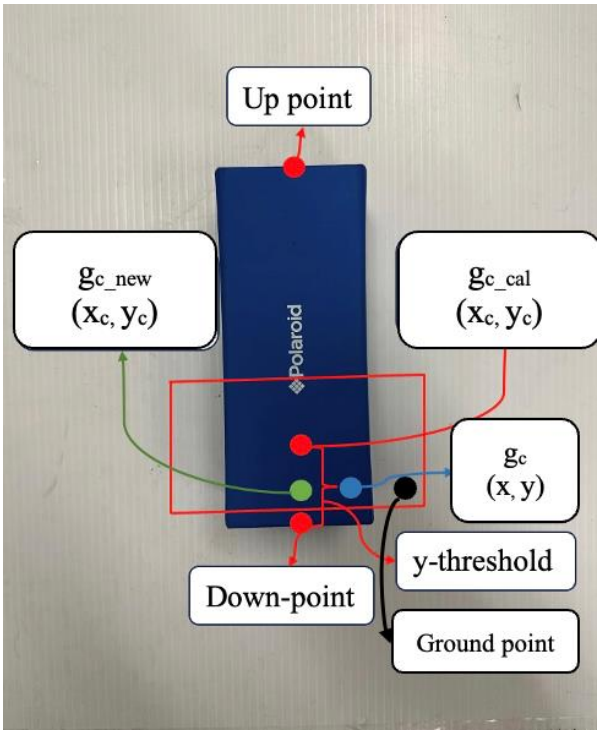


Figure 3.10, the Ground point

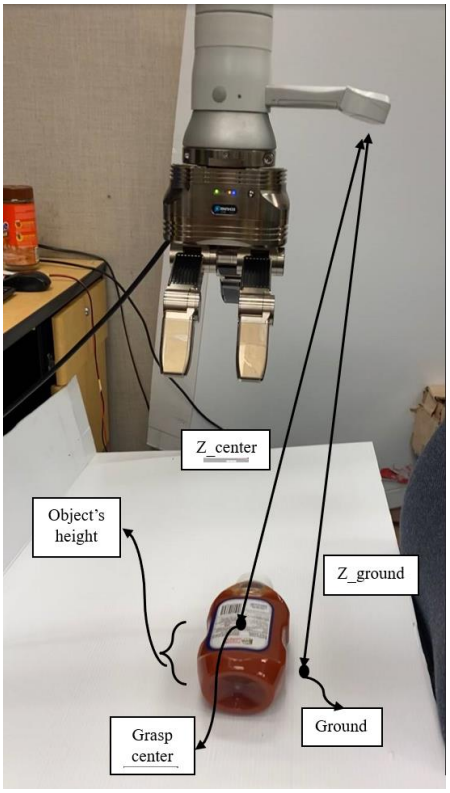


Figure 3.16: Object height definition

3.5.7. Grasp type determination based on grasp width and object's height

As is explained in detail in Section 3.6, we eliminate our grasping options to lateral, tripod, and power grasp (Figure 3.18). Lateral is suitable for grasping smaller objects, tripod grasping is been used for larger objects compared to the lateral grasp. The power grasp is specifically designed for relatively larger objects. It involves creating an enveloping grasp around the object and creating multiple contact points with the object for a more stable grasp. To keep the consistency of Section 3.5 and extracting all the required parameters we define the grasp type Determination in this section. However, a detailed explanation of different grasp types and the strategy of closing fingers are explained in Section 3.6.

As explained in Section 3.6, we narrow down our focus to three distinct grasping options: lateral, tripod, and power grasp (see Figure 3.18). The choice among these grasping techniques is informed by the nature and size of the objects to be manipulated. Lateral grasp proves effective for securing smaller objects. Tripod grasp, on the other

hand, is strategically employed for larger objects in comparison to those suited for lateral grasping. The power grasp is designed for relatively larger objects, creating an enveloping grip around the object, and establishing multiple contact points to ensure a stable and secure grasp. To maintain the coherence of Section 3.5 and to extract all necessary parameters, we introduce the concept of Grasp Type Determination in this section. A comprehensive understanding of the different grasp types and the intricacies of the finger-closing strategy is expounded upon in Section 3.6.

For grasping objects viewing from the side, we have enough space to perform power grasp and hence we use only power grasp because it is the most stable type of grasp. However, when considering the grasping of objects when viewing from the top, the grasping space might be limited and objects can have a variety of shapes and sizes, as a result, the selection of an appropriate grasp type is guided by the following table.

Table 3.1: Defining Grasp type, based on the height of the object in the z direction and width of the grasp

Object	Grasp Width (Section 3.5.5)	Object's height (Section 3.5.6)	Grasp type
Object1	Width < pinch threshold	NA	Lateral
Object2	Width > pinch threshold	Height < power threshold	Tripod
Object3	Width > pinch threshold	Height > power threshold	Power

Let's delve into the explanation of the table's contents:

Object 1: If the width of the object is found to be less than the defined pinch threshold, and the height is not a significant factor in the grasp, the appropriate grasp type to employ is the pinch grasp. In this case, it is recommended to utilize two fingers placed closely together to achieve the desired grasp.

Object 2: When the width of the object surpasses the pinch threshold, while the height remains below the defined power threshold, the optimal choice for grasping is the tripod grasp. The tripod grasp involves utilizing three fingers, strategically positioned to ensure stability and secure gripping of the object.

Object 3: For objects where both the width and height exceed their respective thresholds (pinch and power thresholds), the most suitable grasp type is the power grasp. Employing

the power grasp entails utilizing the entire hand to firmly grasp the object, thereby providing enhanced stability during manipulation.

It is crucial to note that the values of the pinch and power thresholds are predetermined, based on the specific requirements and characteristics of the grasping task. These threshold values are typically derived from experimental evaluation or existing knowledge to optimize the grasp planning for a variety of objects.

Furthermore, in the case of viewing from side grasping, the power grasp is universally recommended as the most stable and reliable type of grasp due to its ability to provide a robust grip and ensure object control from different angles.

By referring to this grasp type determination methodology, one can effectively and systematically choose the appropriate grasp type based on the object's width and height.

3.5.8. Determining the End Effector distance from the grasp center

Another crucial parameter to extract is the end-effector position P_e [x, y, z]. The x and y values are extracted from the calibrated rectangle effector within the camera frame (Section 3.4). The “z” of the end effector is determined based on a specific distance from the object's grasp center. However, this distance varies depending on the type of grasp being executed. Furthermore, the distance also differs when considering viewing from the side and viewing from top grasping scenarios.

Table 3.2 provides a clear outline of the distances between the end effector and the grasp center for different grasping scenarios:

Table 3.2: distance between end-effector and grasp center for different scenarios of grasping.

Grasp view\Grasp type	Pinch grasp	Tripod Grasp	Power grasp
viewing from top	21 cm	20cm	17 cm
viewing from side	-	-	16 cm

Analyzing the table reveals the following:

1. Viewing from top Grasping: For the pinch, tripod, and power grasps in a viewing from top scenario, the respective distances between the end effector and the grasp center are 21 cm, 20 cm, and 17 cm. These values indicate the desired proximity between the end effector and the object, facilitating effective and controlled grasping viewing from the top perspective.
2. Viewing from side Grasping: In the case of viewing from side grasping, denoted as the most reliable type of grasp, only the power grasp is applicable. The distance between the end effector and the grasp center for viewing from side power grasping is 16 cm. This distance is carefully determined to optimize the grasp by ensuring sufficient contact with the object viewing from the side angle.

It is important to emphasize that these distances have been experimentally determined and are specific to the grasping system under consideration. They allow for consistent and accurate positioning of the end effector during grasping tasks, resulting in reliable and successful object manipulation.

By utilizing this information and considering the appropriate distances between the end effector and the grasp center, we can achieve precise and effective grasping in different scenarios, both viewing from the top and viewing from the side.

3.5.9. Summary of Section 3.5

As previously detailed, a set of nine parameters is crucially defined for the successful grasping of an object. These parameters were introduced in Section 3.7, and throughout the subsequent sections, we systematically extract and refine these parameters to ensure reliable and effective grasping.

To begin, we employ the rectangle network to extract the grasp center parameters. These values are acquired by creating a mask of the object through segmentation methods, identifying its edges as explained in Section 3.7.1. The refinement of grasp center parameters in Section 3.7.2 enhances the accuracy and reliability of the grasp representation.

Moving on to Sections 3.7.3, 3.7.4, and 3.7.4, we identify potential contact points between the hand and the object for different joint angles of the pivotal joint. The goal is to determine the best grasp by minimizing the normal value of these contacts. By selecting the pivotal joint angle that results in the lowest total normal difference, we ensure an optimal grasp configuration.

Subsequently, in Section 3.7.5, we define the grasp rectangle based on the chosen contact points. This rectangle serves as a reference frame for grasp planning and execution. Furthermore, in Section 3.7.6, we leverage depth information and a reference point to determine the height of the object. These values play a crucial role in classifying the grasp type, as detailed in Section 3.7.7. Additionally, in Section 3.7.8, we determine the position of the end effector based on the grasp view and grasp type. This comprehensive approach ensures a systematic and informed process for achieving successful grasping. In summary, through a systematic process of parameter extraction, refinement, and calculation, we establish a robust grasp planning approach. The Figure *3.17* shows two grasp rectangles, before and after tuning. As can be seen from the images below, the accuracy of the rectangle increased based on our method. This enables us to represent and transfer the necessary grasp parameters, ensuring successful object manipulation within the desired world frame.

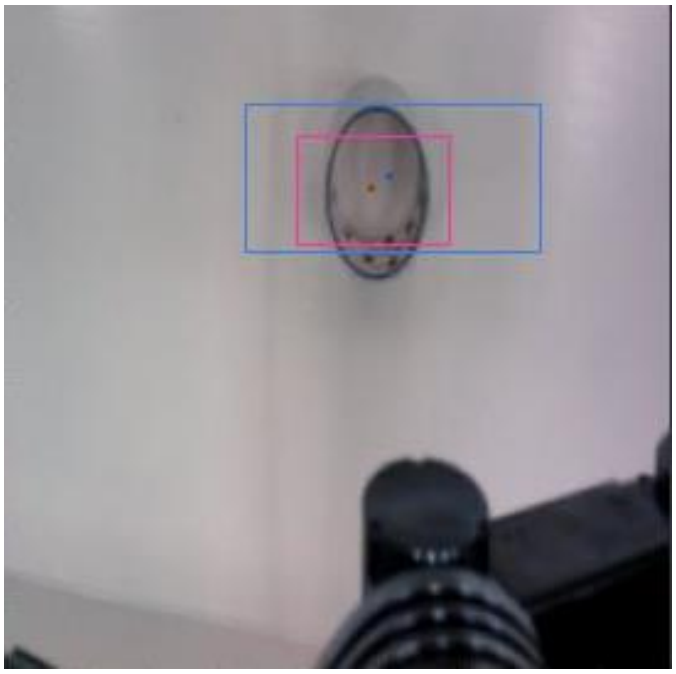


Figure 3.17: The blue rectangle represents the raw output of the network and the pink rectangle shows grasp after calibration in the camera view.

3.6. Different Grasp Types

The goal is to mimic the different grasp types when the fingers touch the object and stop. To grasp objects, the first step is to position the robotic hand near the object at a certain distance (Section 3.5.8), and then we close the fingers based on the size and position of the object. The robotic hand we use consists of a total of 7 DOFs. For simplicity, we have defined three possible grasp types: lateral, tripod, and power grasp (Figure 3.18 and Figure 3.18). As shown in Figure 2.6, the first joints of fingers are named PIP (Proximal joint), and the second joints are named DIP (Distal Interphalangeal). To mimic grasp types, **we need to control the joints' speed** in a way that enables them to touch and grasp the object according to the desired grasp type.

The lateral grasp resembles a parallel jaw gripper, where the distal pads or fingertips move toward each other in a pinching motion. This type of grasp is ideal for handling smaller and thinner objects with precision(Figure 3.18.a and Figure 3.18.b). In the lateral grasp the second joints (DIP), tend to maintain a straighter alignment, approaching zero degrees. In other words, the PIP (First joints) need to move much faster compared to the DIP joint.

In tripod grasping, the DIP joints bend more compared to lateral grasping as can be seen in Figure 3.18.c and Figure 3.18.d. However, the PIP joints still have a higher joint angle and as a result, we close them faster than DIP joints. The tripod configuration allows us to grasp objects of larger size compared to the lateral grasp. This versatility makes it suitable for a wider range of object sizes.

The power grasp is specifically designed for relatively larger objects. It involves creating an enveloping grasp around the object and creating multiple contact points with the object for a more stable grasp. This approach provides a secure and stable grip on the object. We mimic this configuration by allowing the DIP and PIP joints to close at the same speed.

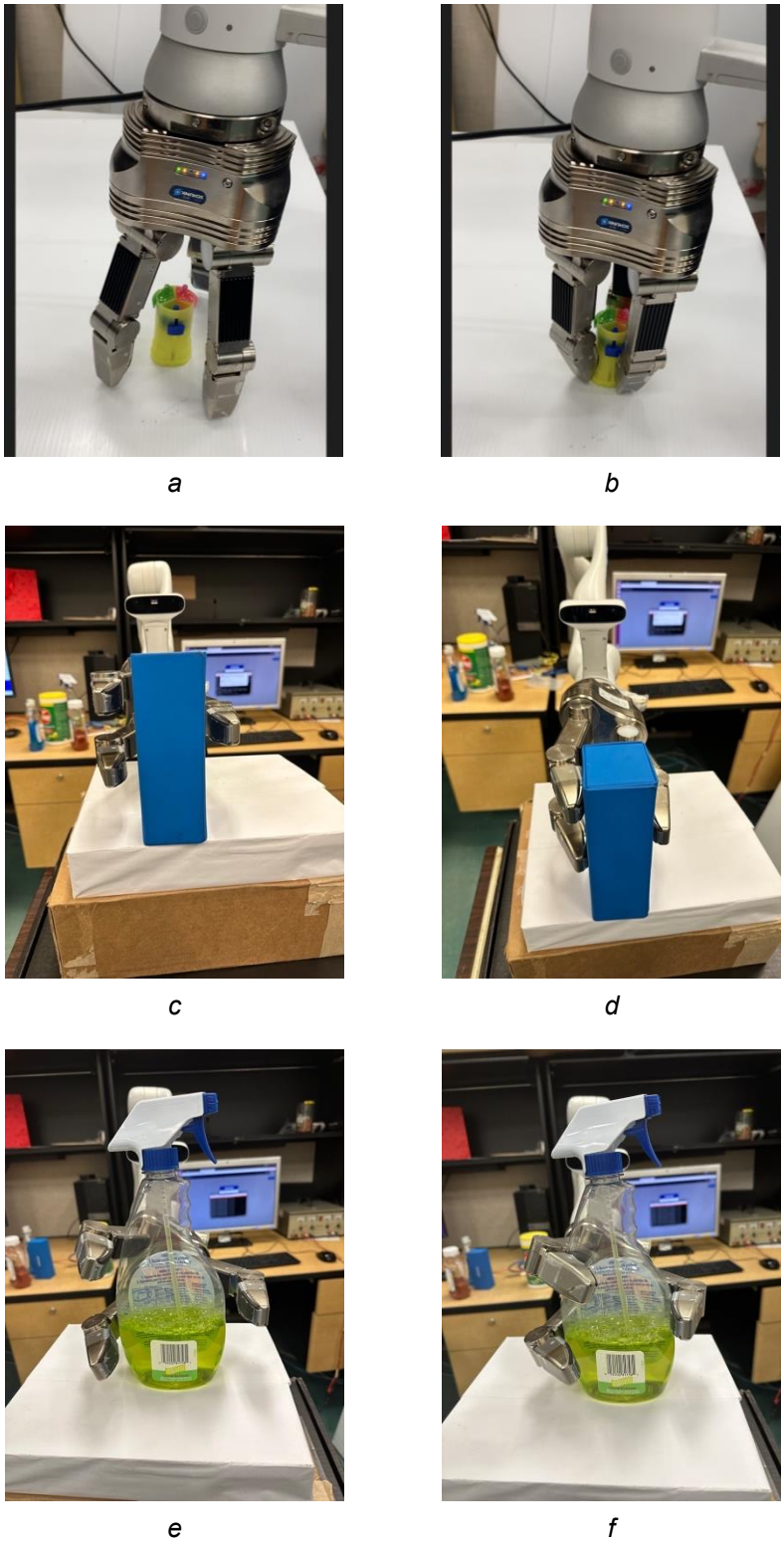
As explained earlier, we regulate the speed at which the fingers close to achieve the desired grasp. The modulation of finger closure speed is managed through a parameter known as "nominal speed" (NS), acting as a setting that influences the rate at which the finger joints come together. The NS parameter signifies the nominal rotational speed of the joints and is adjusted based on the specific grasp type. By multiplying the NS parameter by a specific factor, we can effectively alter the joint speed for each grasping

technique. Table 3.3 outlines the relationship between joint speed and grasp type. For example, in the lateral grasp, the speed of the PIP joints is three times faster than in the power grasp. Similarly, in the tripod grasp, the PIP joints close at a rate two times faster than in the power grasp. Notably, the DIP (Distal Interphalangeal) joint speed remains consistent across all grasp types.

It should be mentioned that parameters have been established through experimentation of the Power, Tripod, and Lateral grasping types, aiming to enhance grasping success rates in each respective scenario. Figure 3.18 shows grasping before and after the grasp.

Table 3.3: joint speed compares to Nominal-Speed (NS) value for each joint

The joint name\ grasp type	Power	Tripod	lateral
PIP (joint 1)	1 x NS	2 x NS	3 x NS
DIP (joint 2)	1 x NS	1 x NS	1 x NS



a *b*

c *d*

e *f*

Figure 3.18: a, c, and e present before grasp and b), d), f) present grasps after execution for lateral, tripod, and power grasp respectively.

3.7. Controlling finger movement and pressure using tactile feedback

Our multi-finger robot employs tactile sensors to detect pressure exerted on objects, a common feature in many similar robotic systems. As shown in Figure 2.6, each fingertip is equipped with multiple tactile sensors. The DIP (Distal Interphalangeal) links are comprised of a 7x9 mesh of sensors, while the PIP (Metacarpophalangeal) links consist of an 8x9 sensor mesh. To simplify the analysis, we utilize the mean pressure from all tactile sensors associated with each fingertip as an indicator of the total pressure applied during manipulation tasks. Also, the sensor data is very noisy so taking means will help with dealing with noise.

During the process of closing the fingers and establishing a firm grip on an object, it is essential to exert sufficient pressure without exceeding the limits that may cause damage to the object or the robotic hand. To address this requirement, we have implemented a control system that adjusts the finger closure speed based on the pressure detected by the tactile sensors. This control system serves as a feedback mechanism, gradually slowing down the speed of finger closure as the pressure reaches specific thresholds, known as DIP_Pressure_limit and PIP_Pressure_limit, respectively. The pressure sensors provide a resolution of 12 bits, meaning pressure values can digitally range between 0 and 4095, corresponding to 0-250 KPa. We have set the DIP_Pressure_limit and PIP_Pressure_limit to 2500 and 2000, respectively.

The primary objective of the feedback mechanism is to ensure a gradual reduction in finger closure speed as the pressure on the object increases. By slowing down the finger movement, we can maintain a delicate balance between exerting enough force to secure the object and preventing excessive pressure that could potentially lead to damage. This approach allows for precise control over the applied force, ensuring a safe and controlled grip on the object.

The integration of this feedback-driven control system significantly enhances the overall performance and safety of our multi-finger robot. By continuously monitoring the pressure through the tactile sensors and adjusting the finger closure speed accordingly, we can achieve reliable and dexterous interaction with objects in various scenarios.

Our closing strategy involves incrementally closing the fingers until reaching the specific pressure threshold assigned to each link (Figure 3.18). If the pressure on the DIP (Distal Interphalangeal) link reaches its limit, we simultaneously halt the corresponding PIP (Metacarpophalangeal) link. However, in the event that the PIP pressure limit is reached first, we halt the PIP link while allowing the DIP link to continue its movement until it also reaches its pressure limit or the predefined time limit elapses.

The underlying rationale for this strategy is twofold. When the tactile sensors on DIP links sense contact and enough pressure, further movement in the PIP joint could amplify pressure at the contact point. To ensure optimal contact with sufficient pressure, the movement is arrested. Conversely, if the PIP link makes initial contact and senses enough pressure, resuming motion in the DIP joint could potentially introduce another contact point without exacerbating pressure at the PIP contact. This enables the finger to establish multiple contacts with the object, which is particularly pertinent in power grasp scenarios.

To prevent indefinite movements, we have imposed a time limit on all actions. The pseudo code provided below outlines the grasp procedure, which has been explained in Sections 3.6.

Algorithm 3

- *Set the grasp type (Power, Tripod, Pinch)*
- *Reset the timer*
- *Start closing the joints based on the speed defined for each Grasp type (Table 3.3)*
- *While (timer < time_limit or all_joints_velocity > 0):*
  - *Calculate the mean pressure for each PIP and DIP links*
  - *For each finger:*
    - *DIP_velocity. = DIP_NS * (1 – DIP_pressure / DIP_pressure_limit)*
    - *PIP_velocity = PIP_NS * (1 – PIP_pressure / PIP_pressure_limit)*
    - *if (DIP pressure >= DIP_pressure_limit):*
      - *DIP_velocity = 0*
      - *PIP_velocity = 0 (Halt stopping the PIP when the DIP_pressure reach to each limit)*
    - *else if (PIP pressure >= PIP_pressure_limit):*
      - *PIP_velocity = 0*
- *Stop*

Chapter 4. Experimental results

Our methodology has been rigorously evaluated using four distinct approaches. 1) we assessed the accuracy of the calibrated grasp rectangle network by conducting experiments on the cornel dataset [41]. This involved measuring the network's ability to accurately predict grasp rectangles.

2) To evaluate the necessity of the post-pressor algorithm, we conduct an experiment to grasp an object with only the output of the GR_ConvNet network and without any calibration.

3) we evaluated the accuracy of the YOLO network, which was employed for object detection. By analyzing the network's performance, we determined its effectiveness in correctly identifying objects within the given context. This method has tested the side view test dataset as it has been explained in Section 3.3 .

4), we assessed the success rate of grasping objects viewing from the top and viewing from the side separately. This involved executing grasping actions and measuring the proportion of successful grasps achieved.

4.1. Grasp rectangle evaluation

As many of the other research in this area [34], [36], [52], we are testing the output of our network and its calibration process, on the test dataset to compare its accuracy on it. To ensure a fair comparison of our results, we adopt the rectangle metric proposed by Jiang et al. [26] to evaluate the performance of our system in grasp detection. This metric defines the criteria for a grasp to be considered valid, which are as follows:

1. The intersection over union (IoU) score between the predicted grasp rectangle and the ground truth grasp rectangle must be greater than 25%.
2. The difference in grasp orientation between the predicted grasp rectangle and the ground truth rectangle should be less than 30 degrees.

After feeding the image into the GR-ConvNet network proposed by [35] doing its calibration process as explained in Section 3.7 and creating the new rectangle, we

compare the results with the grand truth rectangle with the explained method. We used 5-fold cross-validation to evaluate our network performance like the base model.

Table 4.1 shows the accuracy presented by Cornell's official website [41] as three best accuracies reported. Our network is an enhancement on top of the network presented networks.

Table 4.1: Comparing the output of networks

Network name	Accuracy on Cornell grasp dataset
Grasp_det_seg_cnn [36]	98.2
GR_ConvNet [35]	97.7
Resnet50 multi_grasp [52]	96.0
SB_ConvNet (ours)	98.6

Following the application of our proposed calibration process, detailed in Section 3.7, to the base network architecture, SB_ConvNet emerges as a standout performer, achieving an impressive accuracy of 98.6%. This substantiates the efficacy of our calibration technique in enhancing the network's grasp detection capabilities. In comparison, the other networks, Grasp_det_seg_cnn, GR_ConvNet, and Resnet50 multi_grasp, exhibit accuracies of 98.2%, 97.7%, and 96.0% respectively. Notably, our network's accuracy of 98.6% not only surpasses these notable benchmarks but also outperforms the three best-reported accuracies listed on the Cornell official website. This showcases the advancement offered by SB_ConvNet and underscores its potential to enhance the precision and reliability of grasp detection in real-world scenarios.

4.2. Accuracy of YOLO network for detecting objects

After our model finished its training, it showed remarkable results. It achieved an accuracy of over 98.24% on the evaluation dataset and 97% on the test dataset in just under 50 rounds of training. Additionally, we've taken on the task of retraining the Yolov5 model, as mentioned in [53], [54], using the dataset generously provided by [50].

This section is all about thoroughly evaluating how well the retrained Yolov5 model can detect objects. We've looked at it from two different angles: a view from the side and a top-down view. We carefully selected 20 different objects and paired each one with its corresponding pictures. To make the evaluation even better, we made sure to rotate each object in five different ways, which really helped us understand the model's performance.

The model successfully detected 96.0% of the objects in different positions and orientations. It should be pointed out that in our evaluation, we assumed all the objects were standing upright and clearly visible to the camera. This worked well because our robotic hand is designed to grab objects that are standing up straight.

However, we did notice that the model struggled when it came to seeing objects from above. Its accuracy dropped to 73% in this case. This was especially true for smaller objects with different shapes that the model hadn't seen much of during training. Since the grasp rectangle method was showing promising results in creating boxes, we decided to use it for top-view grasping. This method was very applicable to our case, especially for top-down grasping.

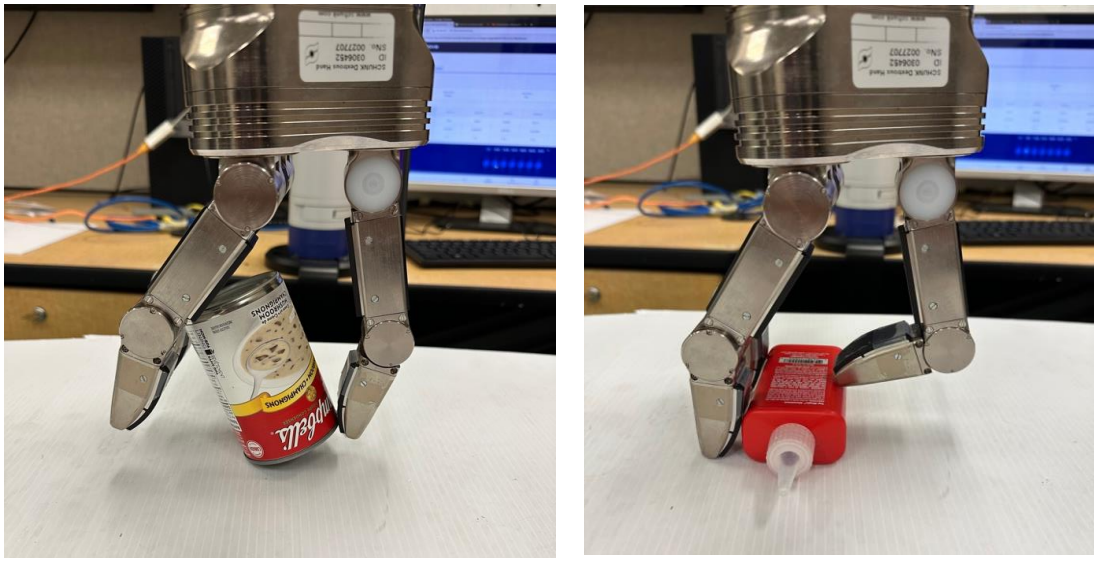
4.3. GR_ConvNet without any post-processing

Each object underwent individual assessment, being presented sequentially before the camera. These evaluations involved varying the object's placement and orientation. It is important to note that prior to evaluating our post-processor function, we first examined the raw output of the GR_ConvNet [35], without applying any post-processing. This preliminary step allowed us to gauge the system's performance and highlight the essential role that the post-processor plays in refining the results. We focused on extracting the necessary parameters, as elaborated in Section 3.7 while excluding the procedures outlined in Sections 3.5.2. It's important to note that we set the pivotal_joint_angle to zero due to limitations in available information. This initial assessment revealed a modest success rate of 65%. We identified three primary reasons for these outcomes:

4.3.1. Unbalanced pressure

Figure 4.1 demonstrates a scenario where uneven pressure was applied to the object, potentially causing its displacement from the intended grasp location or, in more severe

instances, leading to the object falling. Such imbalanced pressure significantly increased the likelihood of an unsuccessful grasp. This highlights the crucial nature of calibrating the x value of the grasp, as outlined in Section 3.7.1.



a) *b)*

Figure 4.1: a) Unequal force will result in the object's fall down and unsuccessful grasp attempt and b) unbalanced pressure will move the object to another position and result in unsuccessful grasping.

4.3.2. Missed Contact

Figure 4.2 illustrates situations where the grasp placement was too close to the object's edges or even outside its boundaries. This often resulted in a finger failing to contact the object, resulting in missed grasps. The importance of accurately calibrating the y value, as detailed in Section 3.4.1, is evident here.

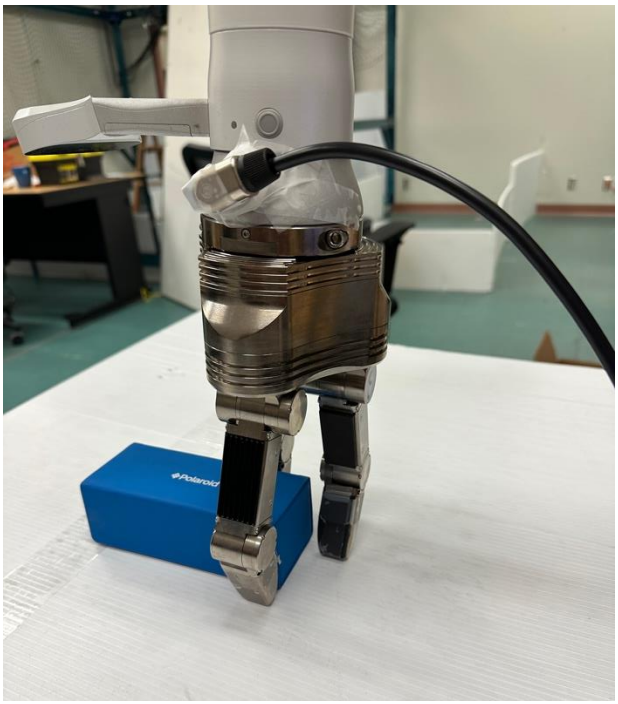


Figure 4.2: Unsuccessful grasp because one finger missed the object while it was closing.

4.3.3. Slippage due to the anchor force

As depicted in Figure 4.3, instances arose where the object slipped from the robot's grip due to the anchor force exerted by the object's center of mass. This highlights the significance of aiming to grasp objects near their center of mass.

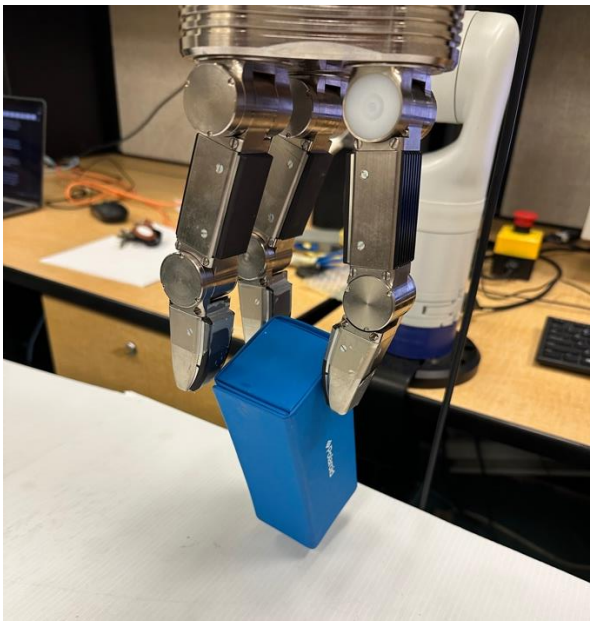


Figure 4.3: Unsuccessful grasp due to the slippage.

In conclusion, the utilization of a postprocessor proved indispensable for the multi-finger robot's successful grasping endeavors, especially when dealing with relatively larger objects compared to those previously grasped using grippers. This evaluation serves to emphasize the critical role of fine-tuning various parameters and employing a comprehensive approach in achieving effective and reliable grasping outcomes.

4.4. Grasping objects viewing from the top and viewing from the side

The evaluation process involved several stages to ensure the successful grasping of objects by the robot. Initially, the robot assumed a scanning pose, allowing it to survey and identify objects within its visual range. Once an object was detected, the robot followed the outlined procedures in Section 3.7 to execute the grasping maneuver.

Our findings underscore the critical importance of fine-tuning the network specifically for multi-finger robotic systems, which significantly enhances the overall success rate. To reinforce this point, we conducted additional tests on 20 objects, with 200 attempts made for grasping viewing from the top, and 15 objects, with 150 attempts, for viewing from side

grasping. These additional tests consistently demonstrated the improvements achieved through network calibration.

Remarkably, our approach achieved an impressive success rate of 94.4% for viewing from top grasping and 95.6% for viewing from side grasping. As depicted in Figure *4.4*, various grasp poses were successfully executed during viewing from top grasping. Furthermore, our method exhibited efficiency, with average calculation times of 0.120 seconds for viewing from top grasping and 0.123 seconds for viewing from side grasping.

A video demonstrating samples of real experiments conducted with our algorithm with our experimental set-up comprising a Gen 3 Kinova arm with a wrist-mounted RealSense D415 RGBD camera and 3-fingered Schunk Dextrous Hand (SDH) is posted at https://summit.sfu.ca/item/38159.

Table *4.2* compares our approach to some of the leading studies in robotic grasping that use CNN models. Our (SB_ConvNet) is a promising choice for successfully grasping objects, especially in real-world scenarios with multi-fingered robots. It's worth noting that for multi-finger robots, we've achieved a significantly higher success rate (95%) than the success rate reported, e.g., [21]in previous work [21] on household objects. Furthermore, our processing time (120 milliseconds) is significantly shorter than the processing time reported (8.1s) in [21].

However, it is important to note that our method encountered challenges when attempting to grasp objects of solid white or transparent colour due to limitations in the grabCut method. Despite this limitation, our approach showcased significant advancements in grasping objects of varying shapes and colours. By implementing a more advanced segmentation method, we are confident that we can overcome this challenge as well.

It's important to note that while we haven't conducted direct comparisons with other studies, there could be variations in processing time and outcomes across different computer setups and processors. Nevertheless, our method demonstrates significant advancements, particularly in the realm of multi-finger grasping techniques. We've observed a remarkable improvement in grasp success rates compared to existing methods in this domain.

Table 4.2: Output of different methods on real-world experiments.

Network Name	Input	processing time(s)	number of fingers	Success Rate (%)
DDGGC [21]	Point-cloud	8.1	3-finger robot	71
Multi-FinGAN [5], [21]	Point-cloud	9.1	3-finger robot	60
Voxel-MDN-Prior [22], [55]	Point cloud	-	3-finger robot	75
Multi-grasp-detector [52]	RGB-D	0.25	Gripper (2)	89
GR-ConvNet [35]	RGB-D	0.2s	Gripper (2)	*95.4*
GR-ConvNet (our test)	RGB-D	0.023	3-finger robot	65
SRConvNet- (ours)	RGB-D(viewing from side)	0.120	3-finger robot	96.0
SRConvNet- (ours)	RGB-D (viewing from top)	0.123	3-finger robot	95.0


NON-CONCENTRATED
NON CONCENTRÉ


KINOVA


KILLS COLD & FLU VIRUSES
DISINFECTING AND CLEANING WIPES
KILLS 99.99% OF VIRUSES & BACTERIA*
THICK & STRONG
LEMON FRESH
75
CAUTION
IRRITANT

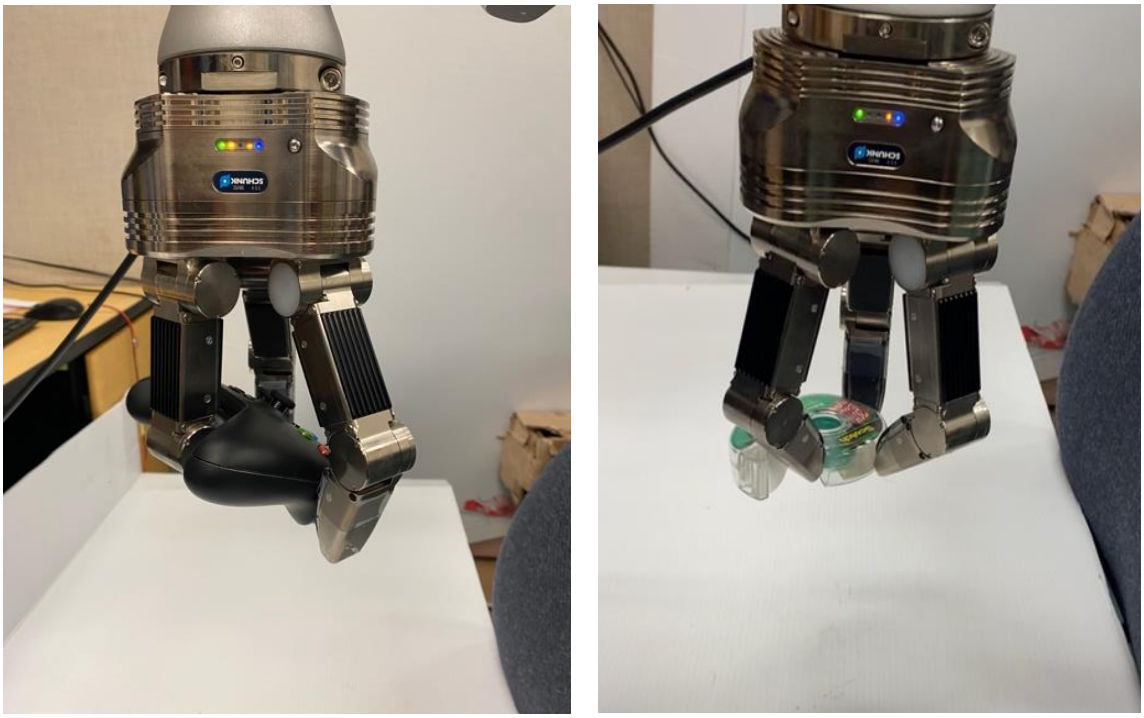


Figure 4.4: Some examples of successful grasps.

4.5. Limitations

In our project, we thoroughly assessed our model across various conditions and scenarios, including comparisons with prior studies (refer to Table 4.2). While we achieved success in accurately grasping household objects, we encountered challenges with objects possessing unique shapes. Some examples of these problematic objects are illustrated in Figure 4.5.

For instance, in Figure 4.5.a, our search-based method struggles due to limitations in determining pivotal joint angles to minimize the distance between the finger direction and the object's normal contact points. This limitation stems from the structure of the SDH hand and the interplay between pivotal joints. Our method, optimized for common-shaped objects, may require adaptation for objects with diverse shapes, possibly necessitating a hand with a different structure and greater dexterity.

Similarly, in Figure 4.5.b, although our method excels in identifying grasp centers and minimizing the angle between finger closing direction and normals at contact points, it may fail to grasp objects from the top due to insufficient contact between fingertips and the object's surface. Such objects may challenge human grasp as well, posing a risk of slipping even from human hands.

a) b)

Figure 4.5: An example of objects that our search based method might fail to grasp

Furthermore, as discussed in Section 3.3, our segmentation approach using the GrabCut method encounters difficulties with white-colored objects, failing to distinguish them from the background. While colorful scenes can aid segmentation in many cases, certain scenarios, as depicted in the accompanying figures, present challenges where the method falters, leading to model failure. Future studies may necessitate a more advanced segmentation model to address these limitations effectively.

Additionally, our model's performance has not been evaluated across different lighting conditions, which could potentially impact its effectiveness in various environments and scenes.

a) b)

Figure 4.6: Some examples of objects that the GrabCut method failed to segment them from the background. This method is sensitive to white colored objects.

Chapter 5. Conclusion and Future Work

In our study, we've introduced an efficient post-processing technique that harnesses a search-based algorithm to enhance the performance of the well-established "grasp rectangle (GR)" method for multi-finger robots. What sets our approach apart is its remarkable speed and reliability compared to existing multi-finger robot grasping methods. Notably, our proposed post-processing technique not only elevates the outcomes of GR Networks but also, improved the reliability of grasping even for two-finger grippers.

In our project, we employed the GrabCut method for object segmentation, a technique that necessitates predefined bounding boxes. These bounding boxes were generated by YOLO, with one set for grasping from the side and another for grasping from the top. Once the object is segmented, we utilize the edges of the object to determine the normals at each edge point. Initially, we fine-tune the grasp center points. Subsequently, by projecting the fingertips onto the grasp plane, we employed a search-based method to identify the optimal grasp pose. Our objective was to ascertain the pivotal joint angle that would minimize the disparity between the fingers closing directions and the object's normals at the contact points. Ultimately, the required parameters to grasp an object (Refer to Chapter 3) were obtained from the information extracted from the tuned grasped rectangle.

While we employed the GrabCut method for segmentation, it excelled in terms of speed and reliability. However, it did face challenges when detecting transparent and white objects. In our future work, we intend to implement more advanced segmentation methods to tackle these challenges and further enhance the efficiency and effectiveness of our approach. Additionally, we aim to expand our method to encompass the simultaneous grasping of multiple objects from a single image.

Moreover, within this project, our grasp operations were confined to objects placed on a flat surface, limiting our end-effector pose to 4 degrees of freedom (DOF). To broaden the scope of our approach, we aim to extend our method to handle objects not constrained to planar surfaces, thereby increasing the potential DOF of grasp pose to 6. Furthermore, it is essential to validate our method using a variety of robotic hands equipped with higher degrees of freedom. A robotic hand with increased DOF can offer enhanced versatility in grasping objects with complex shapes.

Additionally, we must assess our model's efficacy under diverse lighting conditions and scenarios, including those involving shadows. Evaluating our system's performance in the presence of shadows will provide insights into its robustness across a wider spectrum of real-world environments.

In addition, the tactile feedback introduced in Section 3.5.7 currently applies the same force value, close to a maximum range of the sensor to all objects. However, certain delicate objects, such as eggs, require a specific and often gentler touch. In our future work, we aim to extend our method to accommodate grasping objects with varying force requirements. This enhancement will enable our system to adapt its grasp force based on the specific characteristics and needs of each individual object, thereby improving its versatility and applicability across a wider range of scenarios. [56]

References

[1] C. Piazza, G. Grioli, M. G. Catalano, and A. Bicchi, “A Century of Robotic Hands,” *Robotics, and Autonomous Systems Annu. Rev. Control Robot. Auton. Syst. 2019*, vol. 22, pp. 1–32, 2019, doi: 10.1146/annurev-control-060117.

[2] S. El-Khoury and A. Sahbani, “On computing robust n-finger force-closure grasps of 3D objects,” in *2009 IEEE International Conference on Robotics and Automation*, IEEE, May 2009, pp. 2480–2486. doi: 10.1109/ROBOT.2009.5152272.

[3] R. Krug, D. Dimitrov, K. Charusta, and B. Iliev, “On the efficient computation of independent contact regions for force closure grasps,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Oct. 2010, pp. 586–591. doi: 10.1109/IROS.2010.5654380.

[4] R. R. Devaraja, R. Maskeliūnas, and R. Damaševičius, “Design and evaluation of anthropomorphic robotic hand for object grasping and shape recognition,” *Computers*, vol. 10, no. 1, pp. 1–14, 2021, doi: 10.3390/computers10010001.

[5] J. Lundell *et al.*, “Multi-FinGAN: Generative Coarse-To-Fine Sampling of Multi-Finger Grasps,” in *Proceedings - IEEE International Conference on Robotics and Automation*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 4495–4501. doi: 10.1109/ICRA48506.2021.9561228.

[6] R. Newbury *et al.*, “Deep Learning Approaches to Grasp Synthesis: A Review,” *IEEE Transactions on Robotics*, pp. 1–22, 2023, doi: 10.1109/TRO.2023.3280597.

[7] D. Kalashnikov *et al.*, “Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation,” in *2nd Conference on Robot Learning (CoRL 2018), Z¨urich, Switzerland.*, 2018. [Online]. Available: https://goo.gl/wQrYmc.

[8] Q. Lu, M. Van Der Merwe, B. Sundaralingam, and T. Hermans, “Multifingered grasp planning via inference in deep neural networks: Outperforming sampling by

learning differentiable models," *IEEE Robot Autom Mag*, vol. 27, no. 2, pp. 55–65, Jun. 2020, doi: 10.1109/MRA.2020.2976322.

[9] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *J Big Data*, vol. 2, no. 1, Dec. 2015, doi: 10.1186/s40537-014-0007-7.

[10] M. A. Roa and R. Suárez, "Grasp quality measures: review and performance," *Auton Robots*, vol. 38, no. 1, pp. 65–88, Jan. 2015, doi: 10.1007/s10514-014-9402-3.

[11] J. W. Li, H. Liu, and H. G. Cai, "On computing three-finger force-closure grasps of 2-D and 3-D objects," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 155–161, Feb. 2003, doi: 10.1109/TRA.2002.806774.

[12] M. Hegedus, K. Gupta, and M. Mehrandezh, "Efficiently finding poses for multiple grasp types with partial point clouds by uncoupling grasp shape and scale," *Auton Robots*, vol. 46, no. 6, pp. 749–767, Aug. 2022, doi: 10.1007/s10514-022-10049-6.

[13] G. M. Bone, A. Lambert, and M. Edwards, "Automated Modeling and Robotic Grasping of Unknown Three-Dimensional Objects," in *IEEE International Conference on Robotics and Automation*, IEEE Xplore, 2008, pp. 19–23.

[14] M. Hegedus, K. Gupta, and M. Mehrandezh, "Efficiently finding poses for multiple grasp types with partial point clouds by uncoupling grasp shape and scale.," *Auton Robots*, vol. 46, no. 6, pp. 749–767, 2022.

[15] F. Lévesque, B. Sauvet, P. Cardou, and C. Gosselin, "A model-based scooping grasp for the autonomous picking of unknown objects with a two-fingered gripper," *Rob Auton Syst*, vol. 106, pp. 14–25, Aug. 2018, doi: 10.1016/j.robot.2018.04.003.

[16] Siddarth Jain and Brenna Argall, "Grasp Detection for Assistive Robotic Manipulation," in *IEEE International Conference on Robotics and Automation : Stockholm, Sweden*, 2016, pp. 2015–2021.

[17] A. T. Milled, S. Knoopt, H. I. Christensen, and P. K. Allent, “Automatic Grasp Planning Using Shape Primitives.” [Online]. Available: www.cs.columbia.eduramillerigraspit.

[18] Markus Przybylski, Tamim Asfour, and R¨udiger Dillmann, “Planning Grasps for Robotic Hands using a Novel Object Representation based on the Medial Axis Transform,” in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2011, pp. 1781–1788.

[19] M. Nieuwenhuisen, J. Stückler, A. Berner, R. Klein, and S. Behnke, “Shape-Primitive Based Object Recognition and Grasping,” in *7th German Conference on Robotics, May*, May 2012, pp. 1–5.

[20] M. Zhu *et al.*, “Single image 3D object detection and pose estimation for grasping,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2014, pp. 3936–3943. doi: 10.1109/ICRA.2014.6907430.

[21] J. Lundell, F. Verdoja, and V. Kyrki, “DDGC: Generative Deep Dexterous Grasping in Clutter,” *IEEE Robot Autom Lett*, vol. 6, no. 4, pp. 6899–6906, Oct. 2021, doi: 10.1109/LRA.2021.3096239.

[22] Q. Lu, Mark Van der Merwe, and Tucker Hermans, “Multi-fingered active grasp learning,” *IEEE/RSJ International Conference on Intelligent Robots (IROS)*, pp. 8415–8422, 2020, Accessed: May 09, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9340783/

[23] D. Fischinger, A. Weiss, and M. Vincze, “Learning grasps with topographic features,” *International Journal of Robotics Research*, vol. 34, no. 9, pp. 1167–1194, Aug. 2015, doi: 10.1177/0278364915577105.

[24] A. Herzog *et al.*, “Learning of grasp selection based on shape-templates,” *Auton Robots*, vol. 36, no. 1–2, pp. 51–65, Jan. 2014, doi: 10.1007/s10514-013-9366-8.

[25] R. Detry, C. H. Ek, M. Madry, J. Piater, and D. Kragic, “Improving data efficiency of self supervised learning for robotic grasping,” in *Robotics and Automation (ICRA)*, 2019, pp. 3791–3797.

[26] J. Varley, J. Weisz, J. Weiss, and P. Allen, “Generating multi-fingered robotic grasps via deep learning,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Sep. 2015, pp. 4415–4420. doi: 10.1109/IROS.2015.7354004.

[27] T. Boroushaki, I. Perper, M. Nachin, A. Rodriguez, and F. Adib, “Rfusion: Robotic grasping via rf-visual sensing and learning,” in *19th ACM conference on embedded networked sensor systems 2021*, Nov. 2021, pp. 192–205. doi: 10.1145/3485730.3485944.

[28] E. Valarezo Añazco *et al.*, “Natural object manipulation using anthropomorphic robotic hand through deep reinforcement learning and deep grasping probability network,” *Applied Intelligence*, vol. 51, no. 2, pp. 1041–1055, Feb. 2021, doi: 10.1007/S10489-020-01870-6.

[29] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, “A Survey on Learning-Based Robotic Grasping,” *Current Robotics Reports*, vol. 1, no. 4, pp. 239–249, Dec. 2020, doi: 10.1007/S43154-020-00021-6.

[30] I. Lenz, H. Lee, and A. Saxena, “Deep learning for detecting robotic grasps,” *Int J Rob Res*, vol. 34, no. 4–5, pp. 705–724, Apr. 2015, doi: 10.1177/0278364914549607.

[31] M. Mohammed, K. Chung, C. C.-I. Access, and undefined 2020, “Review of deep reinforcement learning-based object grasping: Techniques, open challenges, and recommendations,” *ieeexplore.ieee.org*, Accessed: Apr. 28, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9210095/

[32] D. Kalashnikov *et al.*, “QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation,” Jun. 2018, [Online]. Available: http://arxiv.org/abs/1806.10293

[33] T. Osa, J. Peters, and G. Neumann, “Experiments with Hierarchical Reinforcement Learning of Multiple Grasping Policies,” in *2016 International Symposium on Experimental Robotics*, Springer International Publishing, 2017, 2017, pp. 160–172. doi: 10.1007/978-3-319-50115-4_15.

[34] J. Redmon and A. Angelova, “Real-time grasp detection using convolutional neural networks,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2015, pp. 1316–1322. doi: 10.1109/ICRA.2015.7139361.

[35] S. Kumra, S. Joshi, and F. Sahin, “Antipodal Robotic Grasping using Generative Residual Convolutional Neural Network,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2020, pp. 9626–9633. doi: 10.1109/IROS45743.2020.9340777.

[36] 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)*, IEEE, May 2021, pp. 13452–13458. doi: 10.1109/ICRA48506.2021.9561398.

[37] L. Chen, P. Huang, Y. Li, Z. M.-I. J. of Control, and undefined 2020, “Detecting graspable rectangles of objects in robotic grasping,” *Springer*, vol. 18, no. 5, pp. 1343–1352, May 2020, doi: 10.1007/s12555-019-0186-2.

[38] L. Chen, P. Huang, Y. Li, Z. M.-I. T. on, and undefined 2020, “Edge-dependent efficient grasp rectangle search in robotic grasp detection,” *ieeexplore.ieee.org*, Accessed: Apr. 28, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9311695/

[39] Y. Jiang, S. Moseson, A. S.-2011 I. International, and undefined 2011, “Efficient grasping from rgbd images: Learning using a new rectangle representation,” *ieeexplore.ieee.org*, Accessed: Apr. 28, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5980145/

[40] J. Zhang, M. Li, Y. Feng, and C. Yang, “Robotic grasp detection based on image processing and random forest,” *Multimed Tools Appl*, vol. 79, no. 3–4, pp. 2427–2446, Jan. 2020, doi: 10.1007/S11042-019-08302-9.

[41] “Cornell grasping dataset.” Accessed: Aug. 31, 2013. [Online]. Available: http://pr.cs.cornell.edu/grasping/rectdata/data.php,accessed:2013-09-01

[42] A. Depierre, E. Dellandrea, and L. Chen, “Jacquard: A Large Scale Dataset for Robotic Grasp Detection,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2018, pp. 3511–3516. doi: 10.1109/IROS.2018.8593950.

[43] Kinova Inc, “Kinova Kortex Gen3 7-DOF Robot Arm.” [Online]. Available: https://www.kinovarobotics.com/product/gen3-robots

[44] “Intel ® RealSense TM Product Family D400 Series Datasheet Intel ® RealSense™ Vision Processor D4, Intel ® RealSense™ Vision,” 2022. [Online]. Available: www.intel.com/design/literature.htm.

[45] SCHUNK GmbH & Co. KG, “SCHUNK Dextrous Hand 2.0 (SDH 2.0).” [Online]. Available: http://www.schunk.com

[46] Glenn Jocher, “YOLOv5 by Ultralytics.” May 29, 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[47] R. Huang, J. Pedoeem, and Chen C, “YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers,” in *IEEE international conference on big data (big data)*, Dec. 2018, pp. 2503–2510. Accessed: Apr. 28, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8621865/

[48] W. Fang, L. Wang, P. R.-I. Access, and undefined 2019, “Tinier-YOLO: A real-time object detection method for constrained environments,” *ieeexplore.ieee.org*, Accessed: Apr. 28, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8941141/

[49] T. Diwan, G. Anirudh, and J. V. Tembhurne, “Object detection using YOLO: challenges, architectural successors, datasets and applications,” *Multimed Tools Appl*, vol. 82, no. 6, pp. 9243–9275, Mar. 2023, doi: 10.1007/S11042-022-13644-Y.

[50] P. Douglas De Rizzo Meneghetti *et al.*, “Annotated image dataset of household objects from the RoboFEI@Home team,” Oct. 2020.

[51] M. Bjelonic, “YOLO ROS: Real-Time Object Detection for ROS.” 2018. [Online]. Available: https://github.com/leggedrobotics/darknet_ros

[52] F. J. Chu, R. Xu, and P. A. Vela, “Real-world multiobject, multigrasp detection,” *IEEE Robot Autom Lett*, vol. 3, no. 4, pp. 3355–3362, Oct. 2018, doi: 10.1109/LRA.2018.2852777.

[53] I. S. Isa, M. S. A. Rosli, U. K. Yusof, M. I. F. Maruzuki, and S. N. Sulaiman, “Optimizing the Hyperparameter Tuning of YOLOv5 for Underwater Detection,” *IEEE Access*, vol. 10, pp. 52818–52831, 2022, doi: 10.1109/ACCESS.2022.3174583.

[54] H. Zhang, M. Tian, G. Shao, J. Cheng, and J. Liu, “Target Detection of Forward-Looking Sonar Image Based on Improved YOLOv5,” *IEEE Access*, vol. 10, pp. 18023–18034, 2022, doi: 10.1109/ACCESS.2022.3150339.

[55] Q. Lu, M. Van der Merwe, and T. Hermans, “Multi-Fingered Active Grasp Learning,” Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.05264

[56] A. Tourani, H. Bavle, J. L. Sanchez-Lopez, R. M. Salinas, and H. Voos, “Marker-Based Visual SLAM Leveraging Hierarchical Representations,” in *IEEE International Conference on Intelligent Robots and Systems*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 3461–3467. doi: 10.1109/IROS55552.2023.10341891.

Appendix A.

Video Summary of Thesis and Object Grasping Demonstration

This video provides a summary of the grasping process and demonstrates the robot in action, grasping objects from both the side and the top. It showcases various examples of the robot successfully performing these tasks.

Filename: https://summit.sfu.ca/item/38159

Appendix B.

Files and Code

This link directs to a Google Drive containing all the files and code used and created in this thesis.

Link:
https://drive.google.com/drive/folders/10gTKtssOeb2Z6OVQdz0Cdi3k93sMUfL9?usp=drive_link