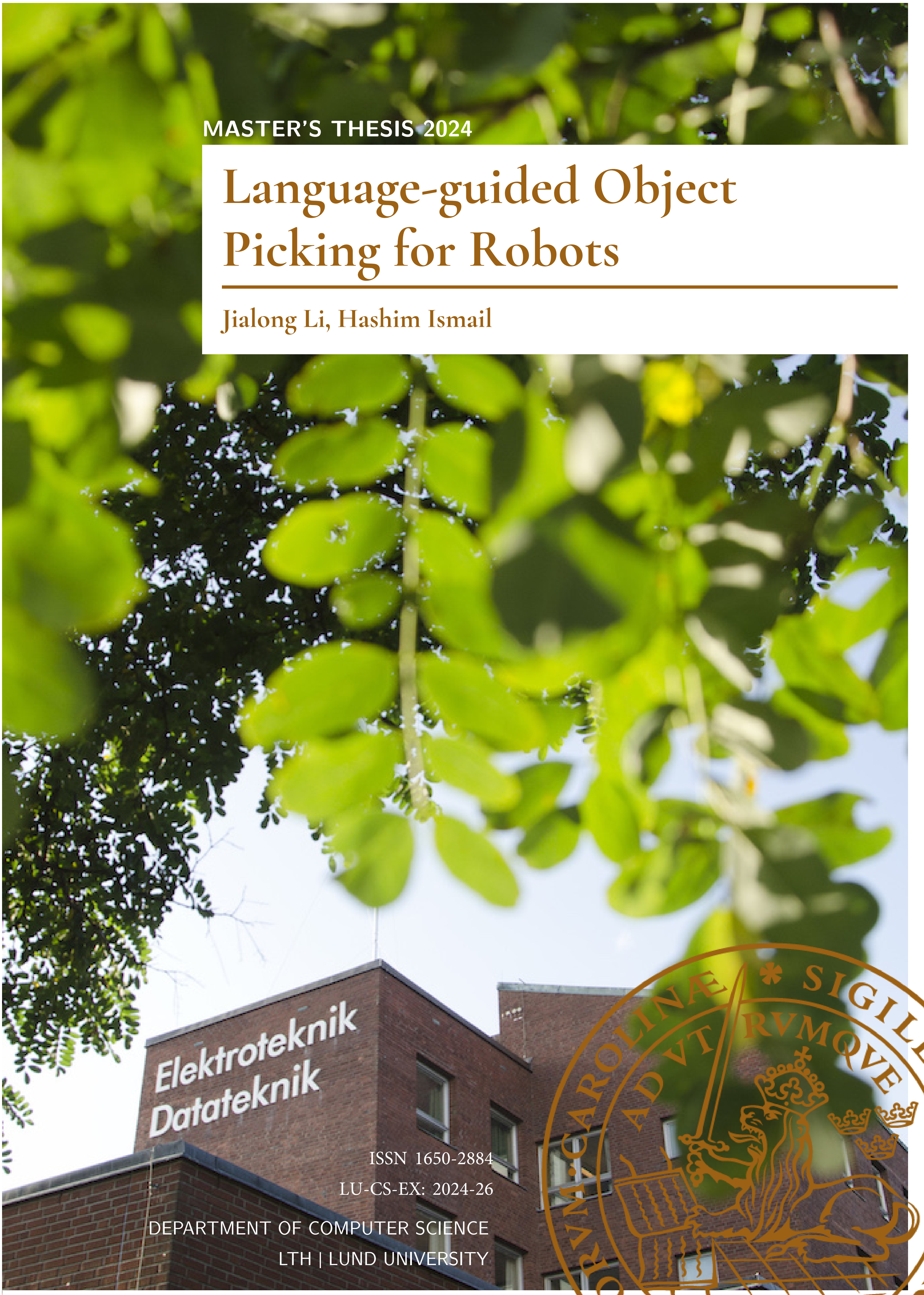


MASTER'S THESIS 2024

Language-guided Object Picking for Robots

Jialong Li, Hashim Ismail

ISSN 1650-2884
LU-CS-EX: 2024-26

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2024-26

Language-guided Object Picking for Robots

Språkförgyllt föremålsplock för robotar

Jialong Li, Hashim Ismail

Language-guided Object Picking for Robots

(A Study on language-guided robot picking from RGB-D)

Jialong Li
ji0341li-s@student.lu.se

Hashim Ismail
ha7172is-s@student.lu.se

May, 2024

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Volker Krüger, volker.krueger@cs.lth.se

Examiner: Michael Doggett, michael.doggett@cs.lth.se

Abstract

The project aims to investigate ways of establishing a deep-learning-based end-to-end pipeline from RGB-D and natural language instructions input to feasible picking, and to apply and validate this pipeline on real robot picking tasks on both causal tasks and specific tasks, identifying its feasibility and limitations.

The pipeline specifically employs the Grounded SAM model for open-set object detection, the Segment Anything model for zero-shot object segmentation, and using either Contact-GraspNet for generating feasible grasps on unknown objects based on 2.5D point cloud inputs, or getting potential picking points by estimating planes of the point cloud inputs

Through testing, it was found that the pipeline performs well in most scenarios, with limitations primarily attributed to the inconsistencies of the vision-language model and the inherent limitations of 2.5D point cloud inputs in representing global scene information.

Keywords: Robotic Grasping, intelligent robot, RGB-D, Vision-Language Model, ROS

Acknowledgements

We would like to express our sincere gratitude to our supervisor Volker Krueger for his invaluable guidance, unwavering support, and expert advice throughout the thesis project.

We are also deeply thankful to Marcus Klang for his invaluable technical support, particularly in the SkiROS2 system, and for his valuable insights and suggestions throughout the project.

Furthermore, we would like to thank Daniel Cederström from Tetra Pak for his continuous follow-up on the project and valuable feedback.

Finally, we commemorate the two mugs that were broken during our robotics experiments.

'Sacrifices must be made.' - Otto Lilienthal

Contents

Part I

Introduction and Theoretical Framework

Chapter 1
Introduction

1.1 Background

Since the birth of robots, researchers have pursued the use of robots to assist human production activities, aiming to enhance efficiency and alleviate the workload of humans. Thanks to the rapid development of data-driven artificial intelligence, the goal of achieving much more intelligent assistance from robots does not seem to be far. Specifically, if one can make a robot "smart" enough to understand the natural language of human users, the old "programming and debugging" way of driving a robot will not be necessary, making robot usage more friendly to non-professional users, and unlocking the potential of natural language instruction.

In recent years, the Large Language Models(LLMs) that Transformers build [27] have shown impressive ability in natural language tasks, such as machine translation and text prediction. Later, Vision Transformer(VIT) [3] further pushed the ability of Transformers into the computer vision field. Sharing similar architecture, the CLIP model (Contrastive Language–Image Pre-training) [22] bridged the gap between text and vision. Following the trend, people started to think of giving the image detection models the ability to classify novel classes, with the assistance of large vocabulary knowledge from LLMs. This is how the open vocabulary vision-language model steps on the stage. Soon, this new branch of the Transformers family tree bore a series of fruits represented by GLIP [11], Dino [28], and Grounding Dino [13].

By adopting open vocabulary model into robot vision systems, the robots can have the potential to become intelligent assistants that can take natural language prompts such as "Pick the cup on the right", and then automatically identify the cup on the right of the scene and process to pick it. However, with the help of open vocabulary detection, the robot is able to find "Which object to pick" by using RGB information. But the problem of "How to pick" is more challenging, and requires extra information.

The depth information, which can be obtained easily along with the depth camera, can fill this gap with rich 3D information. Depending on the specific application scenarios, such information can be processed flexibly, by either deep-learning style or point cloud processing.

In this project, using information from a pure vision source (RGB-D), we aim to discover how cutting-edge deep learning methods can be used to address really robotic or even industrial problems and how robust and reliable they can be.

1.2 Objective

In this project, we aim to achieve automatic detection and picking on two scenarios: one for ordinary household object picking, and one for specific Tetra Pak box picking. The first focuses on a full deep-learning solution, while the latter tries to develop a more robust method as the industrial task normally requires.

1.2.1 Normal Object Picking

To integrate object detection and grasp generation with a full deep learning solution, enabling more flexible and accurate grasping by the robot, this project establishes a pipeline comprising three models:

- **Grounding Dino** [13]: A transformer-based model that is capable of zero-shot natural language-guided object detection on RGB image.
- **Segment Anything** [8]: A transformer-based model generates fine segmentation based on RGB image.
- **Contact-GraspNet** [26]: A PointNet++[21] based model that is able to generate potential grasps from 2.5D point could.

The ideal operation of this pipeline is as follows: given RGB-D input of the scene and prompt natural language input (e.g., "grasp the blue cup") to one end, a feasible grasp set on the desired object is obtained from the other end directly, as illustrated in Figure 1.1.

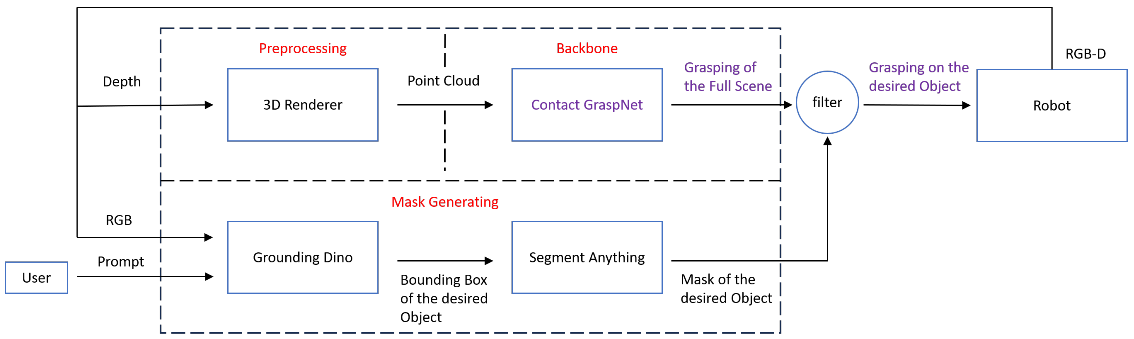


Figure 1.1: The pipeline for normal object picking

1.2.2 Tetra Pak Box Picking

The Tetra Pak box picking pipeline is similar to the previously introduced pipeline as they use the same vision detection system, but differ in the use of depth information. This change is mainly to achieve a more robust system compared to the full deep learning one, as we will introduce more at the Part III. In short, this pipeline contains following part and can be visualized in Figure 1.2:

- **Grounding Dino [13] and Segment Anything [8]** : Same as before.
- **Plane Estimation** : Instead of putting all the depth information into a deep learning model, the point cloud is used to estimate all the potential planes of the scene, then select the upper plane surfaces of all the actual Tetra Pak boxes, and pick up the center point of the plane by suction cup.

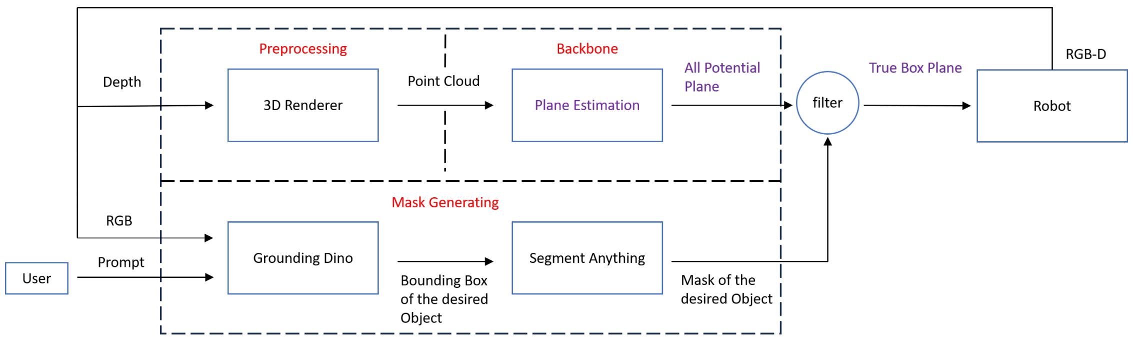


Figure 1.2: The pipeline for Tetra Pak box picking

1.3 Research Question

Based on these two pipelines, in this research we aim to investigate the following questions:

For the vision detection system:

1. How reliable are the prompt-based detection and segmentation models in our tasks?
2. How reliable are the prompt-based detection and segmentation models when specific to Tetra Pak boxes?
3. What form of natural language prompts can better enhance Grounded Dino's accuracy and the grasping process's efficiency?

And for the usage of depth information:

1. How reliable are the grasp poses generated by data-driven methods in our tasks?
2. How reliable are the grasp poses generated by data-driven methods reliable when specific to Tetra Pak boxes?
3. Is the "plane estimation" method better than the deep-learning method when specific to Tetra Pak boxes?

Chapter 2
Related Work

To handle the complexity and uncertainty involved in visual perception, grasp estimation approaches have increasingly turned to data-driven deep learning methods for their adaptability [18]. Within this framework, recent major grasp methods can be sorted into different categories based on how they're trained and used: generative methods, reinforcement learning methods, and direct regression methods.

The generative methods aim to understand the underlying structure of data and recreate similar samples. In work [15], PointNet++ is utilized as the backbone, employing the Conditional Variational Autoencoder structure to encode and decode point clouds for grasp generation. Furthermore, additional features are introduced to annotate the desired point clouds for grasp generation within specified regions. The work of [14]trained a grasp classifier using a generative adversarial network (GAN) and utilized a discriminator to assist in grasping refinement.

Reinforcement learning methods, owing to their inherently task-oriented nature, have been attempted for grasp generation. The work from [7] divided the grasping process into three sub-steps: orienting, approaching, and closing the end-effector. Three reinforcement learning models were separately trained for each step using on-policy learning.

Generative models often require a high ability of the backbone model to capture the full diversity of the data distribution. Additionally, extra grasping post-refinement is typically necessary to filter out unrealistic grasps. Reinforcement learning methods are primarily constrained by their limited generalization capability. This calls for an end-to-end solution, like direct regression methods. Such methods generally aim to achieve a mapping from input information (typically RGB-D images) to one or multiple grasp poses using a single network, that is trained with classical regression. Early work in this field was often inspired by object detection tasks based on RGB images, employing convolutional neural networks for the prediction of top-down view grasping in the image. For example, in the work by [23], the blue channel of the RGB image was replaced with the depth map and AlexNet was applied[9] to achieve a joint task of both object classification and grasping prediction. Recent advancements have been influenced by the development of networks capable of effective three-dimensional structure understanding, such as PointNet [20]. These advancements enable direct estimation of 3D grasping. However, direct regression of 6 degrees of freedom grasping has been found to be a challenging problem [17]. Therefore, many works attempt to reduce the dimensionality of grasping to alleviate the difficulty of network training. The backbone network to be used in this project, Contact-GraspNet [26], belongs to this category of networks. The model addresses the uncertainty issue regarding the three degrees of freedom displacement by generating grasps based on contact points. Further explanation of this model will be provided in chapter 3.

The use of language to assist visual tasks has long been a topic of interest, and the advent of transformers [27] has brought new advancements to this problem. The pioneering breakthrough in this area is the CLIP model (Contrastive Language-Image pre-training) by OpenAI [22]. This model encodes text and images separately using different models and encourages the encodings from different modalities to be as close as possible, with the intention of mapping the encodings of text and images into the same latent space. Benefiting from the powerful encoding capabilities of transformer models and the extensive dataset used for training CLIP, the model performs well on object classification tasks. However, its performance on more advanced visual tasks, such as zero-shot object detection, is less satisfactory. Following in the footsteps of CLIP, the GLIP (Grounded Language-Image Pre-training) model [11] introduces additional text-visual fusion modules, enabling it to achieve good performance on more advanced visual tasks, such as object detection. Building upon the strong foundation laid by these two models, Grounding DINO [13] further enhances text-visual fusion by introducing even more fusion modules and innovative text masking techniques. This results in impressive performance on zero-shot object detection tasks guided by natural language instructions.

By employing the Grounding DINO model, the bounding box of the desired target can be acquired. Following this, grasps generated by the Contact-GraspNet can be filtered, retaining only those within the regions of interest. However, the resulting outcomes may lack refinement, as relying solely on bounding boxes is inadequate for addressing objects with complex shapes or crowded scenes with overlapping objects. Hence, the integration of a segmentation model is necessary to further improve the accuracy of the target detection results. The SAM (Segmentat Anything) model [8] aptly fulfills this requirement. The model achieves prompt-guided segmentation by innovatively encoding the image and the prompt separately, and introducing the prompt into the decoding process. In this project, the combination of Grounding DINO and SAM yields masks of higher precision, enabling refined filtration of grasp results. The specific model fusion approach is informed by the methodology outlined in the work of [24].

Chapter 3
Theoretical Background

The theoretical part comprises three main sections: data preprocessing, which focuses on transforming the input depth image of the scene into point clouds suitable for processing by Contact-GraspNet; the grasping estimation backbone, which delves into the principles behind Contact-GraspNet itself; and mask generating, which explains how detection and segmentation models based on the RGB image of the scene are utilized to generate the mask and filter the output of the grasp model to obtain the desired output. The structure of the theory chapter can be viewed as the red-colored terms denoted in figure 1.1.

3.1 Preprocess

The grasp generation network we employ only accepts input in the form of point clouds. However, in the field of robotics, color and depth images are more common and readily available inputs. Therefore, in our framework, the transformation from RGB-D to point cloud is necessary. Generally, reconstructing point clouds from RGB alone is challenging and requires the use of reconstruction algorithms such as structure from motion [19]. However, this process can be greatly simplified when depth images are available. To explain the specific process, let us begin with how projection is accomplished. The following section is mainly inspired by [5].

3.1.1 Projection with Pinhole Camera Model

A point in the world ($\mathbf{X} = (X, Y, Z) \in \mathbb{R}^3$) can be projected onto a two-dimensional image plane by the camera. One of the fundamental camera models is known as the pinhole camera model, and the process of projection is illustrated in Figure 3.1. It is noteworthy that the intersection point between the principal axis and the image plane is referred to as the principal point $\mathbf{p}$, and the distance from the camera center $\mathbf{C}$ to the principal point is designated as the focal length f [5].

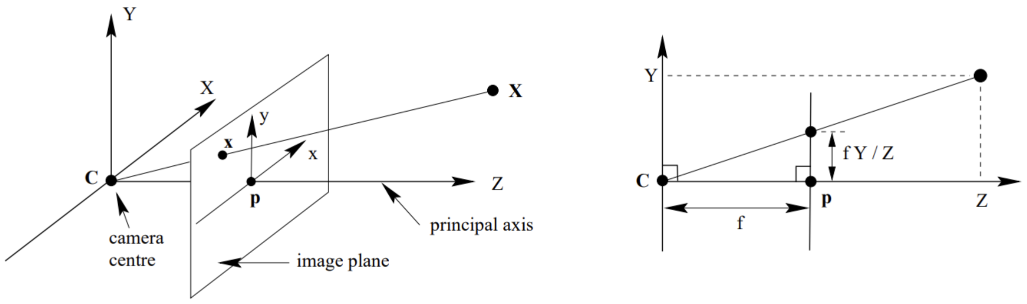


Figure 3.1: Projection via Pinhole Camera, Image from [5], page 154

The image plane, as shown in figure 3.2, includes both the image coordinate system and the camera coordinate system, with an offset (x_0, y_0) from the principal point to the image corner. In most cases the image coordinates are measured in pixels, this introduces the additional effect of unequal scale factors (s_x, s_y in each direction. In total, the projection can be expressed in equation form as equation 3.1 [5], with $f_x = s_x^{-1}f$ and $f_y = s_y^{-1}f$ represents the focal length scaled by the pixel width and height. The matrix $K = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$ is known as the camera calibration matrix or the camera intrinsic matrix. Note the 4-vector $[X, Y, Z, 1]^T$ is the homogeneous representation of the 3-vector world point $[X, Y, Z]^T$.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} s_x^{-1}fX + Zx_0 \\ s_y^{-1}fY + Zy_0 \\ Z \end{bmatrix} = \begin{bmatrix} f_x & 0 & x_0 & 0 \\ 0 & f_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.1}$$

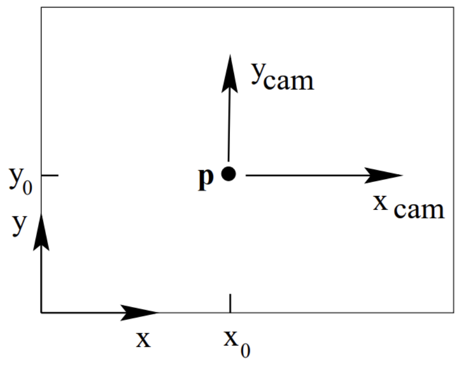


Figure 3.2: Image (x, y) and camera (x_{cam}, y_{cam}) coordinate systems, Image from [5], page 155

The projection described in equation 3.1 occurs under the assumption that the world point (**X**) is described in the camera coordinate system. However, this assumption may not always hold true, as it is more common to express the point in the world coordinate system. In such cases, before applying this projection, an additional transformation is necessary to convert points from the world frame to the camera frame.

Given $\tilde{\mathbf{C}}$ as the coordinates of the camera in the world coordinate system, and R as a 3×3 rotation matrix representing the orientation of the camera coordinate frame, the transformation can be expressed as Equation 3.2, where $\mathbf{X}_{world}$ is also in homogeneous coordinates.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & -R\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \mathbf{X}_{world} \tag{3.2}$$

3.1.2 Depth Image and Point Cloud

A depth image $\mathbf{I} \in \mathbb{N}^{n \times m}$ is a type of image where each pixel stores the distance from the point to the camera, rather than color information, or, in another word, each pixel contains the Z coordinate of the world point $[X, Y, Z]^T$. The process of camera projection, as elucidated in the previous section, involves first transferring world points to the camera coordinate system and then projecting them onto the image plane. The process of reacquiring point clouds using depth images is just to reverse the projecting to image plane part alone, which consists of utilizing the camera intrinsic, restoring the x and y coordinates of each point on the pixels to the camera coordinate system, and adding back the third coordinate z, which is recorded by the depth image. Given the the camera intrinsic matrix as $K = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$, the corresponding 3D point $\mathbf{X}_{i,j}$ of pixel $I(I, j)$ can be computed as:

$$\begin{aligned} Z = I(I, j), X = (I - x_0)\frac{Z}{f_x}, Y = (j - y_0)\frac{Z}{f_z} \\ \rightarrow \mathbf{X}_{i,j} = (X, Y, Z)^T \end{aligned} \tag{3.3}$$

3.2 Backbone

The backbone network, Contact-GraspNet [26], takes in the $2\frac{1}{2}$D point cloud of the whole scene and generates grasp poses at all viable grasping locations. Figure 3.3 illustrates the network's input, output, and inference process. This section will focus on demonstrating the operational principles of the network, as well as its architecture.

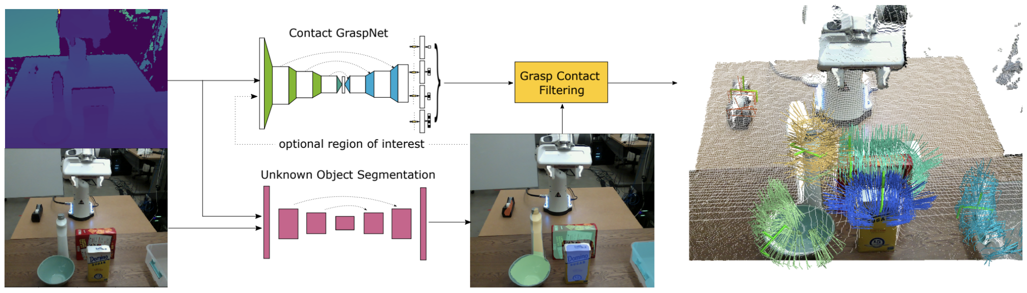


Figure 3.3: Inference process of Contact-GraspNet, picture from [26]

3.2.1 Grasping Estimation

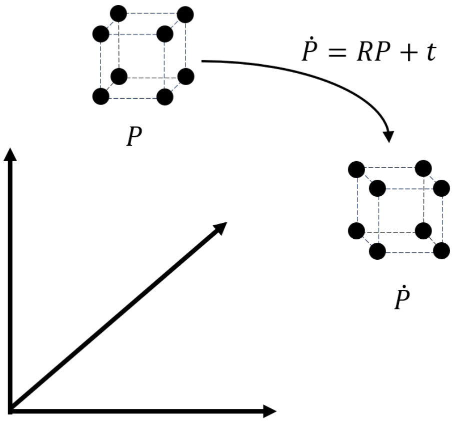


Figure 3.4: Example of transforming a set of points

Before delving into the formal discussion of the network, there is a concept that needs to be clarified: what exactly do we mean when we refer to "grasping." This necessitates an exploration starting from the perspective of similarity transformation matrices.

In three-dimensional Euclidean space, a set of points $\mathbf{P} \in \mathbb{R}^{3\times N}$ are described with respect to relatively coordinate systems. The process of describing the changes in the location of a point set is typically referred to as transformation. Among transformations, similarity transformation is a type of transformation that incorporates translation, rotation, or reflection to the point set while maintaining the relative positions between points. Let the rotation matrix be denoted as $\mathbf{R}_{3\times 3}$ and the translation matrix as $\mathbf{t}_{3\times 1}$. A topical way of transforming a set of points in 3D is as equation 3.4 in homogeneous coordinate, and the idea is demonstrated in figure 3.4.

$$\begin{bmatrix} \hat{P}_{3\times N} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{3\times 3} & \mathbf{t}_{3\times 1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{3\times N} \\ \mathbf{0} \end{bmatrix} \tag{3.4}$$

This is a $SE(3)$ transformation, short for the Special Euclidean group in three dimensions. Expanding on the notion of $SE(3)$ transformation, considering that a set of points can delineate the shape of a gripper, the objective of estimating the grasping can be envisioned as the quest for a series of 6 DoF transformations, which facilitate the adjustment of the gripper by transforming the points to achieve the "appropriate" grasping postures.

3.2.2 Degree of Freedom Reduction

Expanding upon the preceding discussion, in the context of integrating grasp scoring within the same network, traditional grasping endeavors seek to establish a mapping from a point cloud to 6-degree-of-freedom grasp poses alongside their associated scores. This mapping is denoted as $(G, S) = f(\mathbf{P})$, where G denotes the set of potential grasps and S represents the corresponding score. However, direct regression of 6-DoF grasping poses is recognized to be challenging [18], leading many researchers to explore approaches that reduce the dimensionality of the regression problem. In Contact-GraspNet [26], a novel grasp representation was introduced to address this dimensionality reduction task.

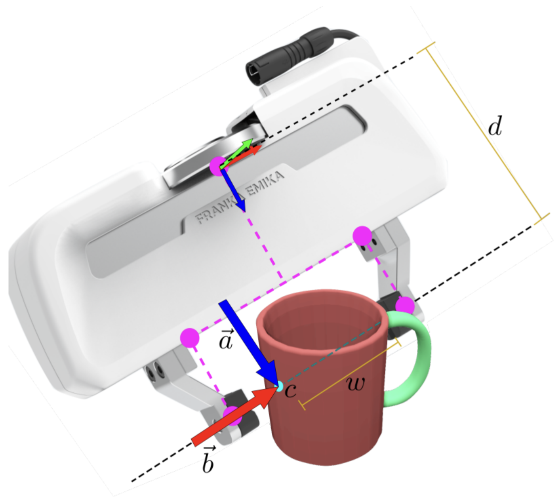


Figure 3.5: Grasp Representation, picture from [26]

Mark the contact point between the object and the gripper as $c \in \mathbb{R}^3$, the grasp can take a new form of representation based on the contact point, as illustrated in Figure 3.5. Herein, $a \in \mathbb{R}^3$ with $||a|| = 1$ denotes the approach vector, signifying the vector from the grip baseline to the object, and $b \in \mathbb{R}^3$ with $||b|| = 1$ represents the grasp baseline vector, indicating the vector from the gripper edge to the contact point. Additionally, $d \in \mathbb{R}$ is the fixed distance from the baseline to the gripper edge, and $w \in \mathbb{R}$ is the unknown grasp width. With these components, a 6-DoF grasp pose $g \in G$ can be expressed using Equations (3.5) and (3.6).

$$t_g = c + \frac{w}{2}b + da \tag{3.5}$$

$$R_g = \begin{bmatrix} | & | & | \\ b & a \times b & a \\ | & | & | \end{bmatrix} \tag{3.6}$$

Equation (3.5) represents the displacement vector reconstructed from the contact point. This formula can be interpreted as a relocation of the origin point from the contact point to the gripper baseline origin point, since the grip serves as the starting frame system for the pose transformation.

Equation (3.6) signifies the reconstructed rotation matrix. Since both a and b are unit vectors and mutually orthogonal, their cross product yields a vector perpendicular to both a and b, thereby enabling the computation of the rotation of the second coordinate as $a \times b$. Furthermore, the network utilizes the in-network Gram-Schmidt orthonormalization when estimating the prediction of a and b (mark as $\hat{a}$ and $\hat{b}$), as depicted in Equation (3.7).

$$\hat{b} = \frac{z_1}{||z_1||} \qquad \hat{a} = \frac{z_2 - < \hat{b}, z_2 > \hat{b}}{||z_2||} \tag{3.7}$$

The Gram-Schmidt orthonormalization process, originally developed for finding an orthonormal basis of a space [1], has been found to be useful in estimating rotations using deep learning. This technique reduces the complexity associated with regressing 3D rotations [29].

The advantage of this novel representation lies in the fact that, in the case of the contact point being known, the remaining unknowns in the equation are the grip width w, the approach vector a, and the grasp baseline vector b. Considering w as a scalar and a and b as part of the rotation matrix, the original 6-DoF search space can be reduced to a more manageable 4-DoF space: 1-DoF for w and 3-DoF for rotation. Considering an extra score added to evaluate the score for grasping, the total degree of freedom is 5. This reduction in dimensionality proves more beneficial to the learning process than approaches that estimate grasping poses in an unconstrained $SE(3)$ space.

However, while the methods described above for representing translation and rotation alleviate the difficulty of grasping regression tasks and are key to the model's convergence and performance, they also become bottlenecks limiting the model's performance. For example, the approach of estimating the second rotation without prior knowledge during rotation matrix reconstruction was found to be unfriendly to the robotic arm with the camera placed on it. Additionally, the assumption that the contact point must be visible and generate grasps solely based on the contact point was found to be easily influenced by the quality of the depth camera. We will discuss these issues in detail in Chapter 5.

Nevertheless, validating this new grasp representation requires a model capable of extracting features from point clouds. Contact-GraspNet employed components of the PointNet++ model [21] to construct such a backbone model. The next chapter will focus on introducing the PointNet++ model.

3.2.3 PointNet++

The Contact GraspNet [26] formed up a feature extraction network that similar to U-Net[25], with its pivotal components drawn from the set abstraction and feature propagation layers introduced in PointNet++ [21].

PointNet++ is a network invented to have a better understanding of the point set that contains points in Euclidean space with the format of (x, y, z) coordinate. In essence, PointNet++'s role in handling point clouds can be likened to the role of convolutional neural networks in image processing. It is a follow-up work of the PointNet [20] network which was designed for the same purpose. To understand what is specific about PointNet++ that makes it a popular backbone for the point set processing task, as PointNet++ is built on top of the previous work PointNet, we begin by introducing how PointNet was designed to address the point cloud feature extraction task.

The points in a point cloud from a Euclidean space usually have such main properties that make it hard to be process like images:

- They are **unordered**. Unlike pixels of color images or voxels in volumetric grids that are naturally ordered with fixed positions, a point cloud contains points that have no specific order. In this case, the network that extracts features from N points of a point set should be robust and invariant to the order of the data being put into the network, which potentially has about $N!$ different orders.

- There exist **interactions** between points. All these points are projected from the real world and together they form meaningful objects, this means none of the points are isolated, each point with its neighbors also forms meaningful subsets. These require the model to capture not only global features but also local structures from nearby points, along with the interactions between local structures.

- The geometric features of the point set are **invariant against transformations**. For example, a point cloud that represents a "chair" should always be a "chair" no matter how the points are rotated and translated together. This required the model to be able to learn the invariant features of the point cloud.

To address such properties of the point cloud, PointNet came up with 3 structures:

- **Symmetry Function Module**: For unordered input, PointNet applied a symmetric function to aggregate the information out of each point. The idea is that the output of the function would not be affected by the order of input points as the function is symmetric. The implementation of such a function is rather simple: this function is approximated by a combination of a multi-layer perceptron network, a single variable function, and a max pooling function, where the symmetry is accomplished by the max pooling function.

- **A local and global information combination Module**: The output from the symmetric function are the global features of the input point cloud. The model has to, as we mentioned before, also be aware of the local structures. PointNet achieved it by a simple process: after the global feature vectors are computed, they are concatenated back to the pre-point feature vectors, and new per-point features are extracted based on such combined features so that the new feature contains both local and global information.

- **Joint alignment network**: To make the network predictions invariant even if some geometric transformations are applied to the point cloud, Pointnet aligns all the input sets to a canonical space before feature extraction, which is done by leaning an affine transformation matrix in the network and applying it directly to the point sets.

Given these three modules, PointNet reached state-of-art on tasks like 3D object recognition or segmentation, but later it was also found that it is not good at learning fine localized details of point cloud. To improve its capability, PointNet++ was designed on top of PointNet and reached even better performance. In PointNet++, the original PointNet was employed as a plug-in module, and two main new structures as well as the two components utilized within the Contact-GraspNet [26] model are the set abstraction and the feature propagation layers.

Set abstraction

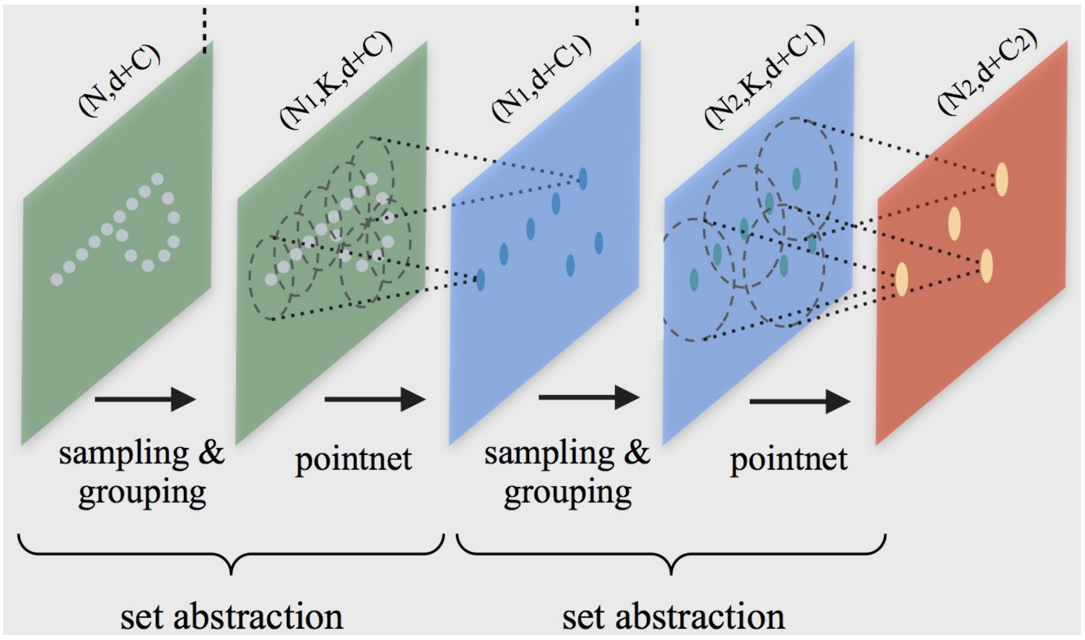


Figure 3.6: The set abstraction of PointNet++, picture from [21]

The Set Abstraction section, as shown in Figure 3.6, plays a crucial role in sampling the input point cloud and effectively capturing local features. This layer significantly contributes to hierarchical feature learning by carefully selecting a subset of representative points and aggregating pertinent information within their local neighborhoods. The following provides a detailed description of the layers within the set abstraction section and their functions, along with figure 3.6 indicate how set abstraction works:

- **Sample Layer:** The initial step of the Set Abstraction layer involves selecting a representative subset of points from the input point cloud. Farthest point sampling (FPS) is commonly employed, where points are iteratively chosen based on their maximal distance from those already selected. These chosen points act as centroids, delineating local regions.

- **Grouping Layer:** The Grouping layer takes the sampled representative points obtained through farthest point sampling (FPS). For each representative point, a local region is defined by considering neighboring points within a specified radius. The grouping operation aggregates points within these local regions, effectively creating clusters of points around each representative point. This grouping enables the network to capture fine-grained spatial relationships and local features. The output of this layer is mini groups of point sets, with each group corresponding to a local region containing a certain number of points in the neighborhood of centroid points.

- **PointNet Layer[20]:** In the final step, a lightweight version of the PointNet network is applied to the local point sets, functioning as a local feature learner. The layer employs shared multi-layer perceptrons (MLPs) to learn point-wise features, capturing local geometric details. These point-wise features are then aggregated across all points using symmetric functions like max pooling, ensuring permutation invariance to the order of points. This permutation invariance is crucial for handling unordered point clouds. The result is a feature vector representing the entire set of points. This layer is applied hierarchically to capture features at different scales, contributing to the overall hierarchical feature learning process.

Within the framework of PointNet++, the Set Abstraction layer operates hierarchically and is iteratively applied at various levels of the network. This hierarchical structure enables the model to capture features across multiple scales, from nuanced fine-grained details to broader global structures. The iterative downsampling and local feature aggregation at each level contribute synergistically to the network's proficiency in comprehending complex 3D scenes.

The Set Abstraction plays a role analogous to the downsampling segment in U-Net [25], acquiring information-rich features through sampling. Similar to U-Net, once features are sampled, they need to be restored to the original scale through upsampling, elucidating the necessity of the Feature Propagation section.

Feature Propagation

The Feature Propagation layers within PointNet++ play a crucial role in upsampling and propagating features back to the original scale of the input point cloud after downsampling in the Set Abstraction layers. These layers receive downsampled features associated with sampled representative points and their respective local regions. Through interpolation techniques like nearest-neighbor interpolation, the features are distributed back to the original points. This process ensures the preservation and integration of hierarchical information learned at different scales into the final feature representation. By concatenating these features with the original point features, the Feature Propagation layers facilitate the combination of fine-grained details captured by Set Abstraction with the global context. This allows the network to comprehend both local structures and larger global patterns within the point cloud. The hierarchical application of Set Abstraction and Feature Propagation layers contributes to the overall effectiveness of PointNet++ in capturing intricate 3D spatial information.

By leveraging the two PointNet++ components introduced above, Contact-GraspNet constructs the network architecture as depicted in Figure 3.7. As previously mentioned, it is an architecture that performs downsampling and upsampling of point clouds to extract features. The feature vectors are then processed by four feedforward neural network heads, with each generating one of the four predictions: grasp score $\hat{s}$, approach vector $\hat{a}$, grasp baseline vector $\hat{b}$, and grasp width $\hat{w}$. The network is trained using supervised learning, and the loss function employed will be introduced in the next section.

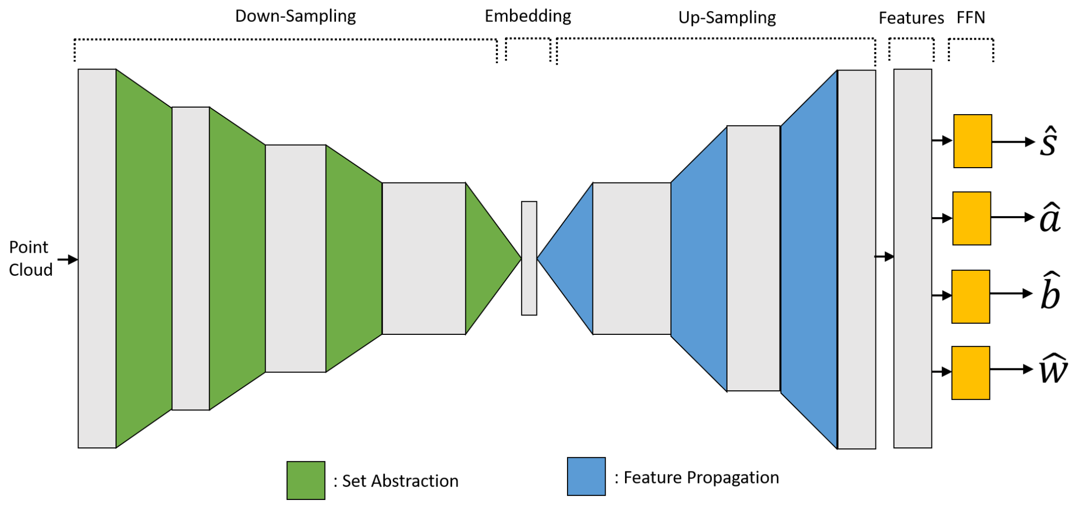


Figure 3.7: The brief overview of Contact-GraspNet

3.2.4 Supervised Learning of Grasp

Assuming at least one visible contact point for all feasible grasp poses, simplifying the pose estimation problem hinges on initially identifying the contact point. Consequently, the total objective of generating grasping can be delineated into two distinct tasks and trained together: contact point classification and pose regression. But before diving into the details, the dataset used for training is worth mentioning.

Dataset

Contact-GraspNet was trained on the Acronym dataset [4], which comprises 1.7 million grasps for 8,872 everyday objects represented in 3D mesh format. During training, objects along with their corresponding ground truth grasps were placed on a plane in simulation, followed by sampling of different camera viewpoints to generate RGB-D information for the objects. Furthermore, the RGB-D images were rendered into 2.5D point clouds. Denote the points $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \in \mathbb{R}^3$, each point in the point cloud was labeled using Equation 3.8.

$$s_i \begin{cases} 1 & \min_j \|\mathbf{p}_i - \mathbf{c}_j\|_2 < r \\ 0 & otherwise \end{cases} \qquad \forall i = 1, 2, \ldots, n \tag{3.8}$$

With $c_j \in \mathcal{P}$ are the mesh contact points of non-colliding ground truth grasp, and r represents the maximum propagation radius. The primary purpose of this equation is to assess, within a radius r around each point, the presence of viable ground truth contact points. If such points are found, a score of 1 is assigned to the respective point. Following this process, the original point cloud is segregated into two subsets: $\mathcal{P}^- := \{\mathbf{p}_i | s_i = 0\}$ and $\mathcal{P}^+ := \{\mathbf{p}_i | s_i = 1\}$.

Point Classification

As discussed in the previous section, in the ground truth $2.5D$ point cloud, each point is assigned a label indicating whether the point is a potential contact point for generating grasps. To learn this label, a binary cross-entropy loss is established as shown in Equation 3.9.

$$l_{bce} = -\frac{1}{n^+} \sum_{i}^{n^+} s_i \log(\hat{s}_i) + (1 - s_i) \log(1 - \hat{s}_i) \tag{3.9}$$

Where $n^+ = |\mathcal{P}^+|$ represents the size of $\mathcal{P}^+$, $\hat{s}_i$ is obtained by applying the softmax activation function over the output of the feed-forward network, as shown in Figure 3.7. The fundamental concept underlying this approach is that by learning the distribution of scores, the model becomes adept at effectively assigning scores to contact points, facilitating contact point classification.

Having classified the contact points, the next objective is to map the pose distribution to these identified contact points through pose regression.

Pose Regression

Recall equation 3.5 and 3.6, the grasping is re-estalished from vector $\hat{a}$ and $\hat{b}$, which is estimated by the network, as Figure 3.7. Another loss is essential to measure the distance between correct grasping and predicted grasping to regress this learned transformation to the contact points. Denoting the resulting estimated rotation and displacement corresponding to point $\mathbf{p}_i$ as $\hat{R}_{g,i}$ and $\hat{t}_{g,i}$ (alongside the ground truth $R_{g,i}$ and $t_{g,i}$), and defining five 3D points $\mathbf{v} \in \mathbb{R}^{5\times3}$ to represent the 6-DoF gripper pose, as illustrated in Figure 3.5, the transformed pose with both predicted and ground truth transformations is defined as shown in Equation (3.10).

$$\mathbf{v}_i^{gt} = \mathbf{v}R_{g,i}^T + t_{g,i} \qquad \mathbf{v}_i^{pred} = \mathbf{v}\hat{R}_{g,i}^T + \hat{t}_{g,i} \tag{3.10}$$

To facilitate the regression of $\mathbf{v}_i^{pred}$ toward $\mathbf{v}_i^{gt}$, a weighted average distance loss is defined, as shown in Equation (3.11).

$$l_{add-s} = -\frac{1}{n^+} \sum_{i}^{n^+} \hat{s}_i \min_u \|\mathbf{v}_i^{pred} - \mathbf{v}_u^{gt}\|_2 \tag{3.11}$$

Here, the variable u signifies that the ground truth $\mathbf{v}_u^{gt}$ is selected among all possible ground truths within the maximum propagation radius r of a given point. The selected ground truth is the one minimizing the distance the most.

Additionally, the grip width w is a parameter that also requires prediction, as shown in Figure 3.7. Contact-GraspNet chose to bin the continuous width and learn it as a label regression task. The loss function, as in Equation 3.12, is a multi-label cross-entropy loss, with M is the number of label, w_{ic} is the symbolic label (1 for sample i belongs to class c, 0 otherwise), $\hat{w}_{ic}$ is the predicted probabilities based on observations obtained by the model through the softmax function.

$$l_{width} = -\frac{1}{n^+} \sum_{i}^{n^+} \sum_{c=1}^{M} w_{ic} \log(\hat{w}_{ic}) \tag{3.12}$$

Finally, the overall loss function for training the model is defined as illustrated in Equation (3.13).

$$l = \alpha l_{bce} + \beta l_{add-s} + \gamma l_{wdith} \quad \alpha = 1, \beta = 10, \gamma = 1 \tag{3.13}$$

3.3 Object Detection and Segmentation

This section introduces the detection and segmentation models used in the pipeline. We utilize Grounding DINO [13] an open-set detection model to generate bounding boxes and extract labels for each object. Then we use Segment Anything Model (SAM) [8] to generate segmentation masks for the detected objects. We first introduce the mechanism of Grounding Dino: a model inspired by GLIP's feature fusion block, and DETR's decoder method.

3.3.1 Detection Transformers (DETR)

DETR [2] is an end-to-end object detection model with an encoder-decoder architecture based on transformers. It uses standard implementations of Transformers [27] and ResNet [6] backbones from standard deep learning libraries.

The original DETR model adopts a simple architecture as shown in figure 3.8. It contains three main components: a standard Convolutional Neural network (CNN) backbone to extract a compact feature representation, an encoder-decoder transformer, and a simple feed forward network (FFN).

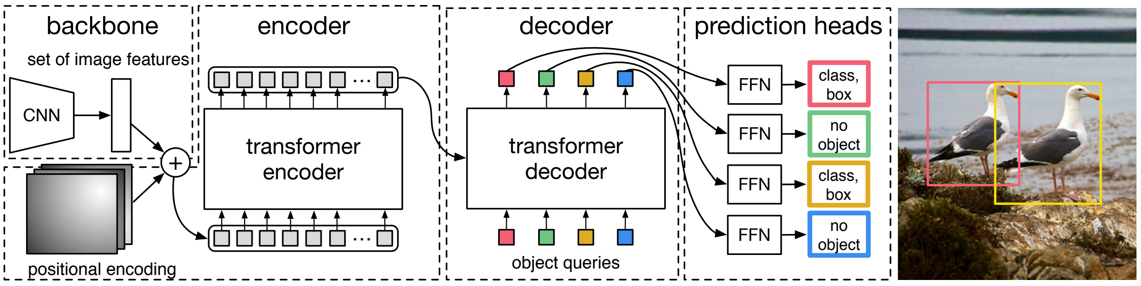


Figure 3.8: DETR Architecture, figure from [2]

DETR decoder outputs a fixed-size N set of predictions in a single pass where N is significantly larger than the number of objects in the input image. The decoder follows the standard transformer decoder architecture by Vaswani [27] with the difference being that DETR decodes the N objects in parallel at each decoder layer, while the original architecture uses an autoregressive model that predicts the output sequence one element at a time.

DETR with Improved DeNoising Anchor (DINO)

Inspired by many follow-up papers that improve on the original DETR architecture, most notably DAB-DETR [12] and DN-DETR [10], the DINO [28] model was introduced. DINO is a DETR-like model that contains a backbone, a multi-layer Transformer encoder, a multi-layer Transformer decoder, and multiple prediction heads. Like DETR, the model extracts multi-scale features with backbones like ResNet or ViT and feeds them into a Transformer encoder with corresponding positional embeddings. In the decoder layer it introduces denoising training as in DN-DETR by feeding noise-added ground-truth labels and boxes into the decoder and training the model to reconstruct them. DINO also improves on query selection in the decoder by selecting the top K encoder features from the last encoder layers as priors to enhance decoder queries.

3.3.2 Grounded Language-Image Pre-training (GLIP)

In the natural language area, the phrase "Ground" is to the process of connecting the model's output to verifiable sources of information, which is what the GLIP model is trying to do: find an efficient way of connecting image information to verified language. Based on CLIP's [22](CLIP (Contrastive Language-Image Pretraining) attempt of grounding image to text by contrastive learning and only fuse the features at the decoder part, GLIP builds their model upon one trial idea: contrastive learning and also fusing the feature before the decoder block.

Figure 3.9 shows their architecture, in the deep fusion part, the feature that comes from text and image are fused three times, with each time the features are encoded again. GLIP's improvement compared to CLIP proves this "Deep feature fusion" is useful for the quality of the language-image pre-training model, and greatly inspires the feature fusion block of Grounding Dino.

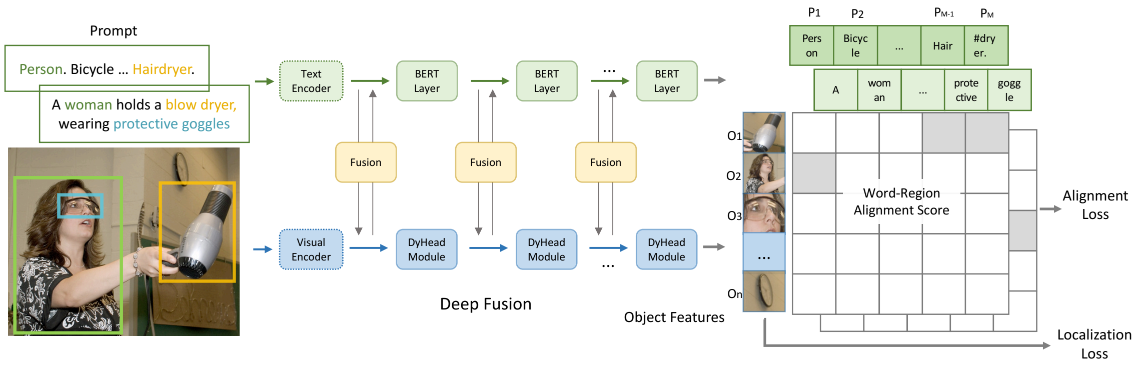
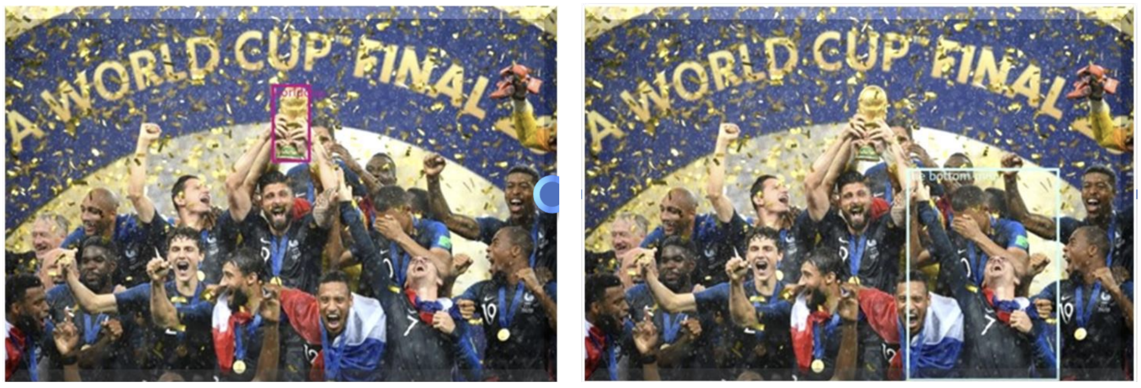


Figure 3.9: GLIP Architecture, figure from [13]

3.3.3 Object Detection With Prompt: Grounding DINO

Grounding DINO [13], inspired by DINO [28] and GLIP [11], intent to achieve language-guided object detection on RGB image. Grounding DINO not only excels in natural language-guided object detection but also possesses two critical capabilities that make it particularly suitable for application in robotic grasping tasks: open-set object detection and referring object detection. An example of both is shown in Figure 3.10.



(a) 'Worldcup'

(b) 'The bottom man with his head up'

Figure 3.10: Left: Openset Detection, Right: Referring Detection, Figures from [13]

These functionalities of the model, especially referring object detection, are well-suited for applications in robotic settings. This is because the workspaces of robotic arms are typically confined, and when faced with unknown objects, the use of descriptive language for recognition has the potential to improve grasping efficiency.

However, the model's output is only accurate up to the bounding box of the object, which may not suffice for robotic grasping tasks that require precision, especially when dealing with objects of complex shapes. In this project, the Segment Anything model is introduced to enhance the results of object detection and obtain accurate masks for the objects.

Grounding DINO consists of a dual-encoder-single-decoder architecture. The model architecture is shown in figure 3.11. First, the vanilla features are extracted from each image and text prompt pairs separately by an image backbone and a text backbone. Then a feature enhancer is used to update both vanilla features using cross-modality feature fusion. Cross-modality queries are selected from image features using language-guided query selection and fed into a cross-modality decoder to generate desired features from the two modal features. The output queries are used to predict object bounding boxes and extract corresponding labels.

Feature Enhancer

To generate the vanilla features, image features are extracted using a ViT image backbone, and text features are extracted using a transformer encoder. These features are then fed into the feature enhancer, as Figure 3.11 shows.

As described in section 3.3.2, Grounding Dino's feature enhancer is built based on GLIP's feature fusion mechanisms. To achieve deep feature fusing, the dual-direction cross-attention is applied to vanilla features after self-attention of both the text features and the image features.

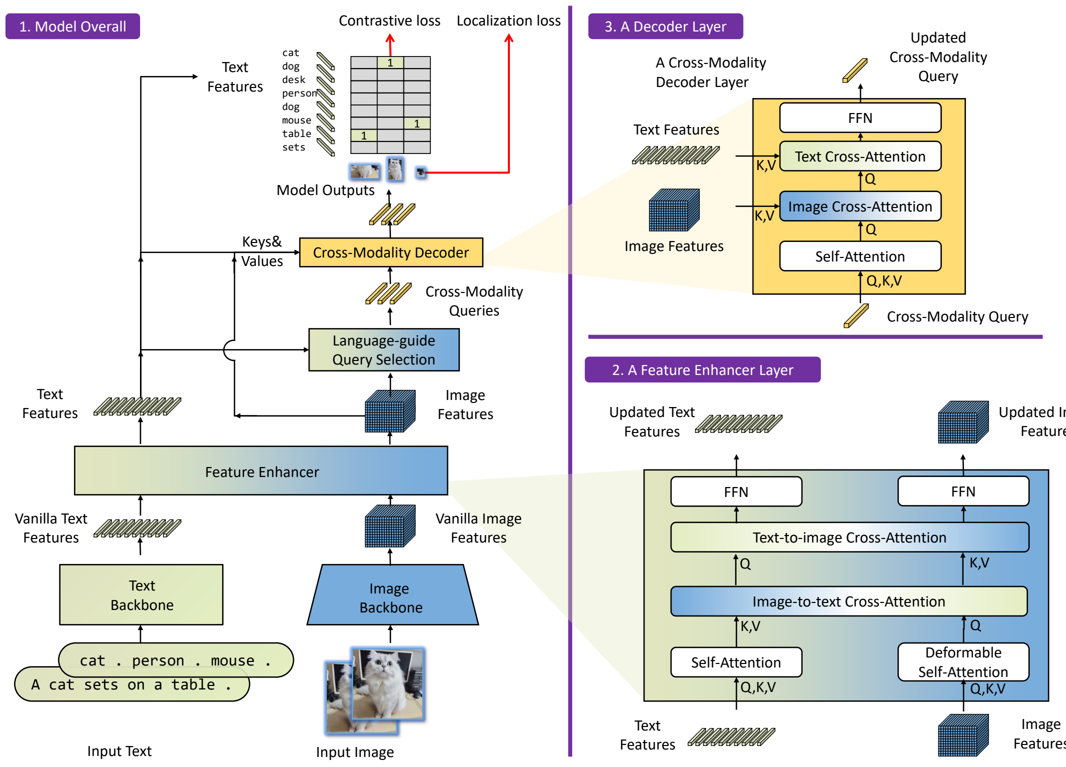


Figure 3.11: Grounding DINO Architecture, figure from [13]

Cross-Modality Decoder

Finally, image and text modality features are combined in the cross=modality decoder as shown in figure 3.11 block 3. Each query goes through a self-attention layer, an image cross-attention layer to combine image features, a text cross-attention layer to combine text features, and an FFN layer in each cross-modality decoder layer. Compared to the original DINO model mentioned in 3.3.1, each decoder layer has an extra text cross-attention layer to inject text information into queries for better modality alignment.

3.3.4 Semantic Segmentation: Segmentation Anything

The Segment Anything model [8], leveraging its powerful zero-shot segmentation capability, excels at segmenting objects that have not appeared in the training dataset [8]. The overview of the model is as Figure 3.12, which contains three blocks:

- **Image encoder**: A pre-trained Vision Transformer (ViT) [3] is adapted to encodes input with even high resolution.
- **Prompt encoder**: Box prompt(as bounding boxes given by DINO) is encoded as position encoding that is transformed and represented by learned embeddings.
- **Mask decoder**: A Transformer decoder block is applied to decode both image embeddings and prompt embeddings by dul-direction cross-attention (image-to-prompt and

prompt-to-image).

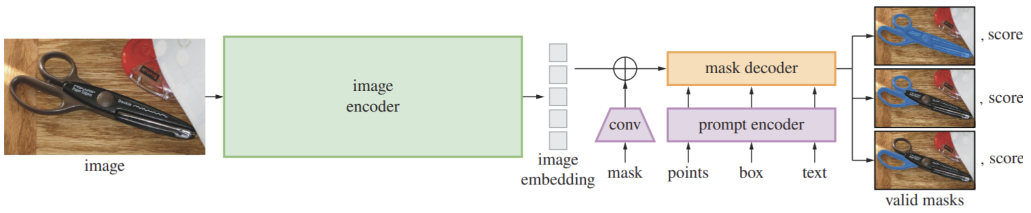


Figure 3.12: Segment Anything Model (SAM) overview, figure from [8]

By combining SAM and Grounding Dino, passing the detection result of Dino as a box prompt and passing it in SAM, a powerful prompt-to-segmentation tool for unknown objects can be achieved, as shown in Figure 3.13.

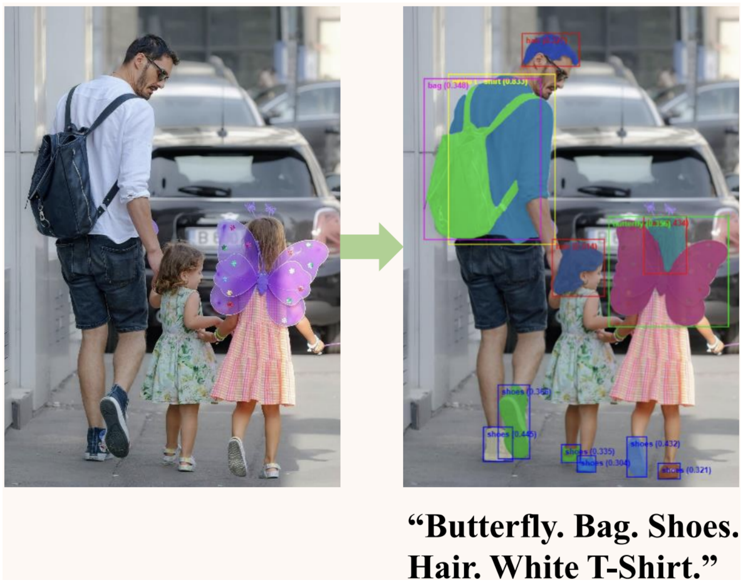


Figure 3.13: Detection and Segmentation with Grounding DINO and Segment Anything together, picture from [24]

The implementation of Grounding DINO and Segment Anything together is mainly inspired by the work of [24].

Part II

Implementation on Normal Object Picking

Chapter 4
Experimental Setup

This chapter focuses on presenting the application of our workflow in robotics, as well as the methods used to drive the robot accordingly.

4.1 The workspace

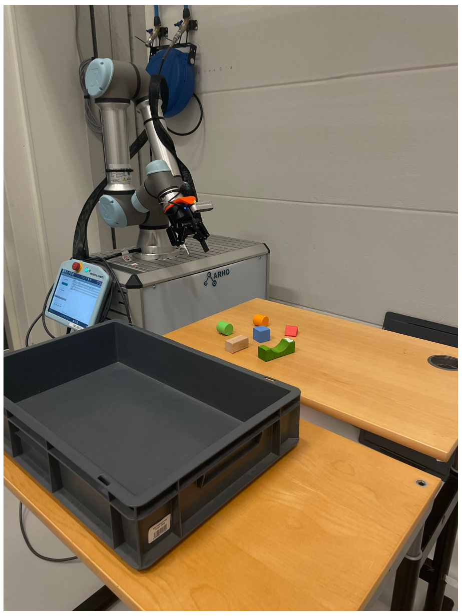


Figure 4.1: Our Workspace

The workspace, depicted in Figure 4.1, comprises the robot arm, a table for staging objects to be picked, and a container situated adjacent to depositing picked objects.

4.2 The Robot

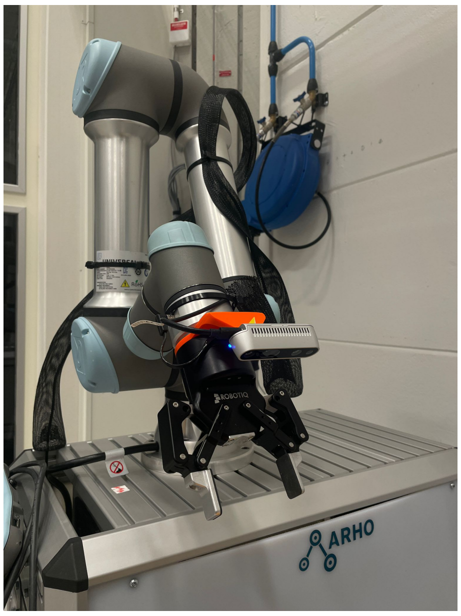


Figure 4.2: The Heron robot

The robot itself (we name it Heron), comprises the following components:

- Intel® RealSense™ D400 camera
- ROBOTIQ® 2F-85 adaptive gripper
- UR5e collaborative robot arm
- MiR100 mobile base

To enable the Heron robot to execute various commands, we utilized the SkiROS2 [16] platform within the Robot Operating System (ROS) to drive its operations.

4.3 SkiROS2: Skill-based Robot Operating Platform

The SkiROS2 framework, deeply interwoven with ROS, is meticulously engineered to manage robot behaviors through modular components [16]. Based on this platform, all robot behaviors are abstracted into individual instructions called "skills." These skills can then be compiled sequentially to drive the robot, facilitating its operation in a structured and organized manner. This tool proves particularly advantageous in environments with a substantial foundational understanding, and it boasts several notable characteristics:

- Global model: Presented as a resource description framework (RDF) database, this structure serves as a universal model for data interchange across the Web. It holds valuable environmental insights that the robot can leverage.

- Robot behavior designer: SkiROS streamlines behavior design via Python programming, empowering the robot to adapt to changes seamlessly during execution.
- Skill system: This system facilitates the organization of programmed behaviors, fostering modularity and reusability. Within SkiROS2, behavior trees are aggregated into entities known as skills, incorporating single or multiple nodes. Each skill integrates pre- and post-conditions that validate execution status at any given moment. The implementation of skills can be structured into Python packages, which can be deployed and imported into other robots, functioning akin to plugins.

4.4 Essential Skills

Built upon SkiROS2, there are several skills crucial for ensuring the proper functioning of the robot and preventing errors during experiments, despite not being included in the global blueprint of our workflow. Specifically, these skills encompass camera calibration, hand-eye calibration, and robotic arm motion planning.

- **Camera calibration**: This skill ensures that the camera's intrinsic and extrinsic parameters, as mentioned at 3.1.1, are accurately calibrated, enabling precise perception of the environment.
- **Hand-Eye Calibration**: This skill aligns the robot's hand with its vision system, thereby ensuring precise coordination between perception and manipulation tasks by accurately transforming between the gripper frame and the camera frame.
- **Motion Planning**: Once the grasping position is determined, this skill plans feasible trajectories for the robot arm to safely and efficiently navigate to the desired grasping location within the workspace.

Chapter 5
Result and Discussion

In this chapter, we carry out picking and placing experiments on the robot to evaluate the process pipeline. Then we discuss the results and outline our findings. The entire evaluation section attempts to assess the capabilities of the pipeline in three different tasks:

1. **Single-object grasping** : Aimed at evaluating the fundamental grasping ability.
2. **Multi-object unguided grasping**: Intended to assess the pipeline's ability to filter background information through segmentation and accomplish grasping in multi-object scenarios without specific guidance.
3. **Multi-object guided grasping**: Testing the overall performance of the pipeline when presented with multi-object scenarios and directional prompts.

The whole process is evaluated on the Heron robot (as described in the last chapter). In all experiments, we execute the highest-scored grasp pose out of the generated grasps. Before, the prompt-guided part, We chose the object with the highest confidence score out of the detected objects using the Grounded SAM model.

5.1 Single Object grasping

In this experiment, we execute the picking task 10 times on each of 14 different objects and mainly show some of the representative results. We aim to test the Contact GraspNet model's ability to consistently and repetitively pick a single object, for objects with both simple and complex shapes. Note all the results are presented in image format. The left side of each image depicts the object to be tested for grasping along with its mask. The right side displays all feasible grasps generated by the model in point cloud form, with grasps marked in green representing those with the highest scores generated by the model. The successful rate can be shown in table 5.1.

Number of Trials	Fail on Detection	Fail on Grasping Estimation	Success
140	2	24	114

Table 5.1: Result of single object grasping

The results for simple-shaped objects include symmetric rectangular and cylinder-shaped objects, as shown in Figure 5.1, as well as long objects and cube-shaped objects depicted in Figure 5.2. For complex objects, we tested non-uniform and non-symmetric shapes, as shown in Figure 5.3, and household objects such as a cup or even a broken cup piece, illustrated in Figure 5.4.

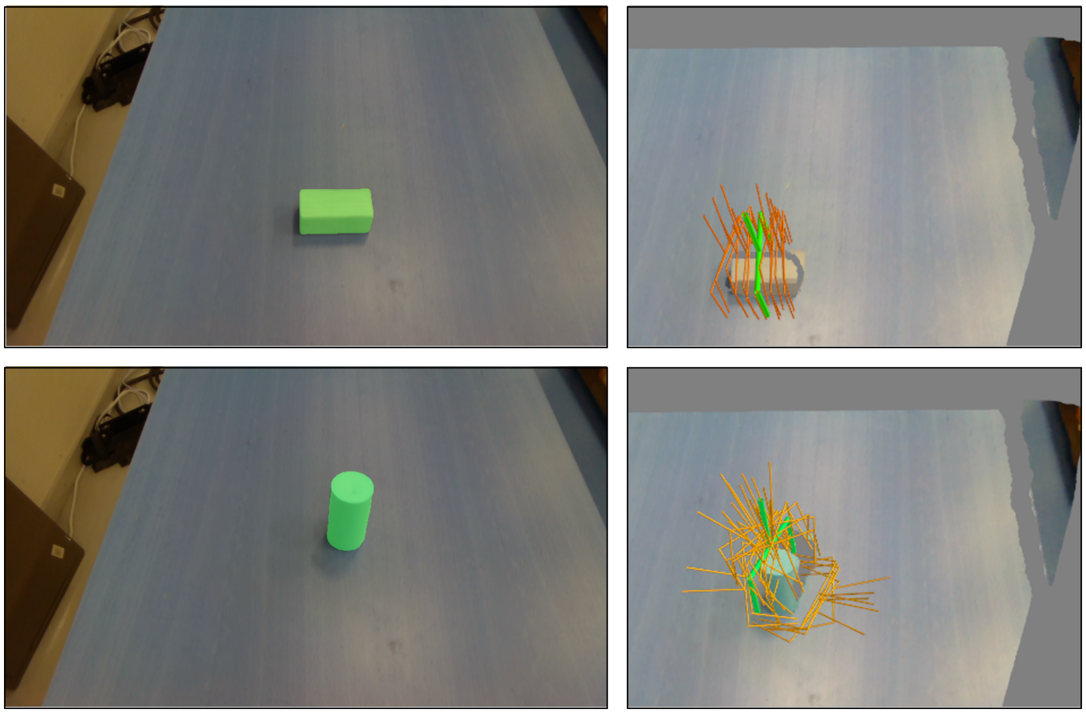


Figure 5.1: Single object grasping for objects 1 and 2

For the results of single-object grasping, we have the following observations, which mainly include one advantage of Contact-GraspNet and two limitations:

- **Advantage**: Contact-GraspNet demonstrates robustness and flexibility in generating feasible grasps for various shapes and sizes of objects, as shown in Figure 5.1 to 5.4, showcasing its versatility in handling unknown objects.
- **Limitation 1**: The model only learns to generate grasps based on the shape represented by the point cloud, without considering physical characteristics such as gravity or the center of mass of the objects. As a result, it may generate "feasible but impractical" grasps, particularly when dealing with heavy objects, as shown in Figure 5.3, the model generates possible grasps all over the body of the object, but the highest-scored grasp is not close to the center of mass of the object.
- **Limitation 2**: The grasps generated by Contact-GraspNet may be constrained by the resolution and quality of the partial point cloud data. As discussed in the previous

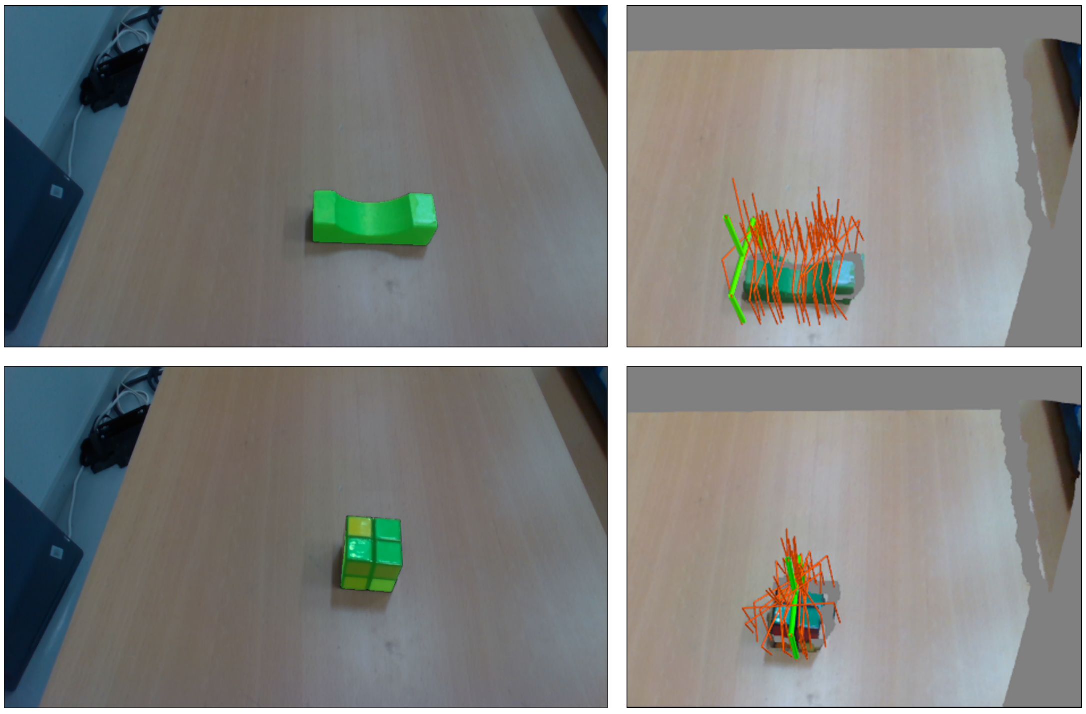


Figure 5.2: Single object grasping for objects 3 and 4

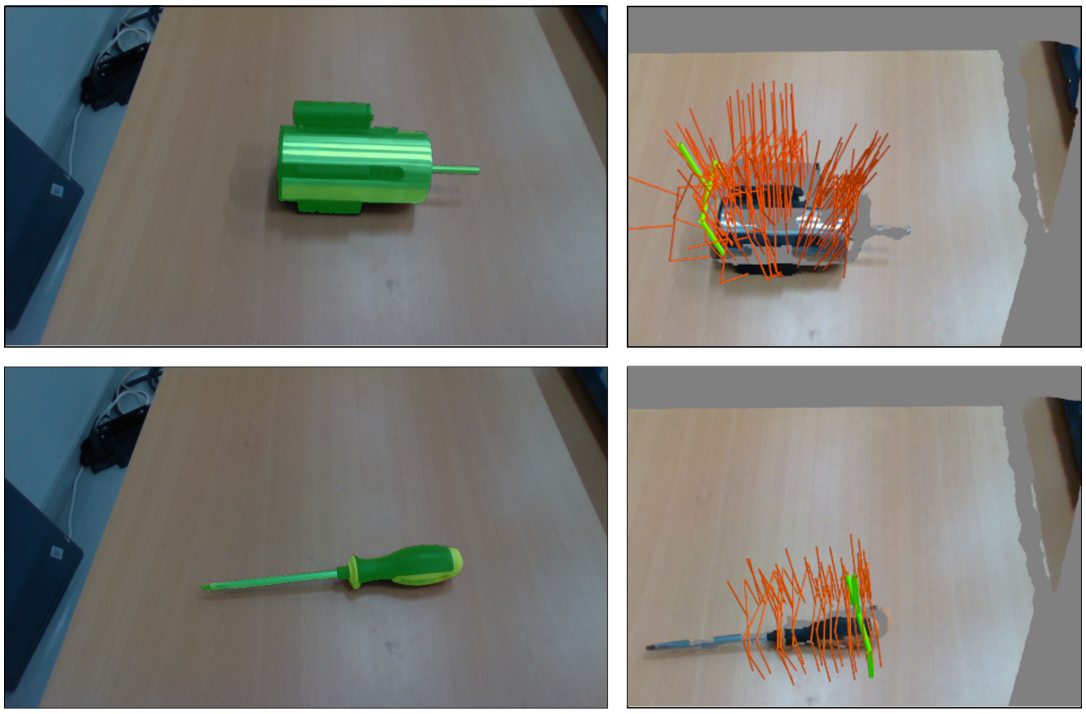


Figure 5.3: Single object grasping for objects 5 and 6

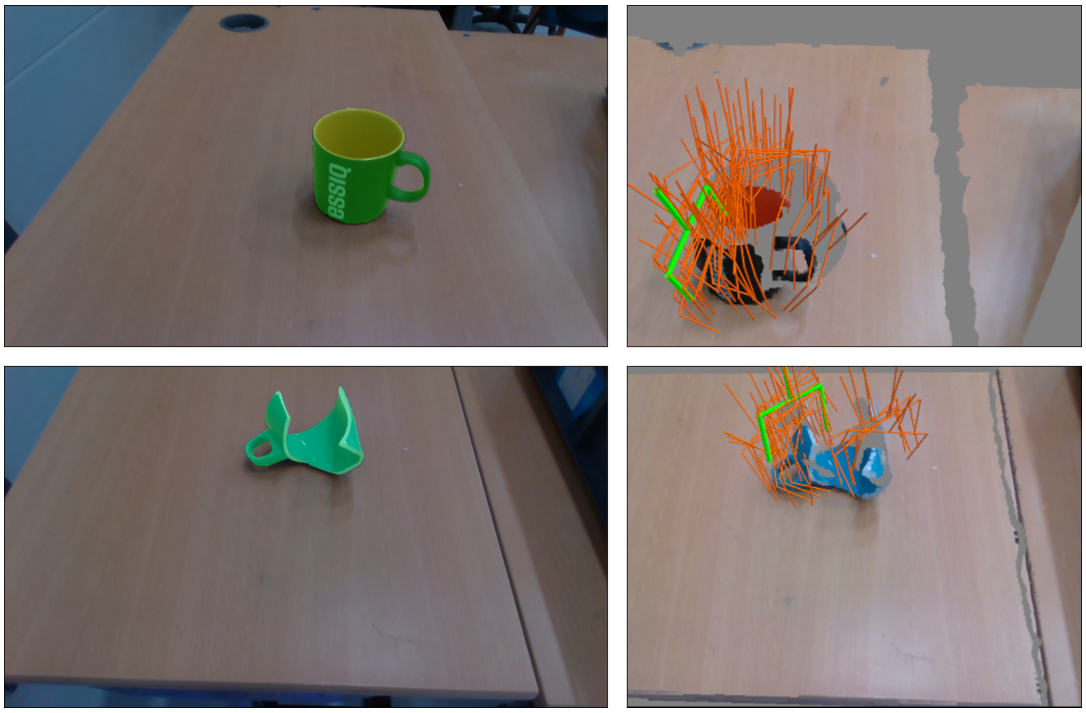


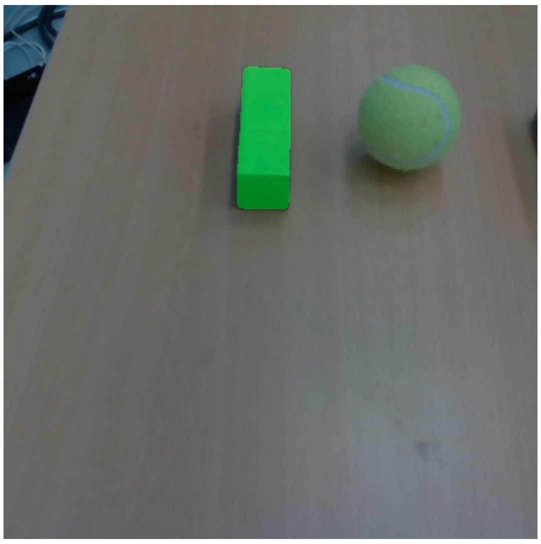
Figure 5.4: Single object grasping for objects 7 and 8

chapter, the model samples points from the given point cloud to generate possible grasps. The partial point cloud consists only of the points the camera can see. Therefore, sometimes when certain areas of the object are not visible to the depth camera, the model can only generate grasps within the visible regions, thus leading to suboptimal grasping. This can be observed initially in Figure 5.4. An example shown in Figure 5.5 illustrates this more clearly. In this Figure, the object is placed in this orientation where the camera can see only the top and front faces of the object, all the points on the side view of the object are not visible in the partial point cloud. The model is then forced to sample a point from either the top or front view of the object. Due to the contact points on the top view resulting in grasps that collide with the table, those samples are rejected. The only contact points left are the ones in the front view. Due to the object being too long for the gripper's width, the only grasps that do not result in a collision will lead to an unsuitable angle for grasping as predicted by the model and seen in figure 5.5c showing the grasp on the object and the contact point highlighted as a green dot.

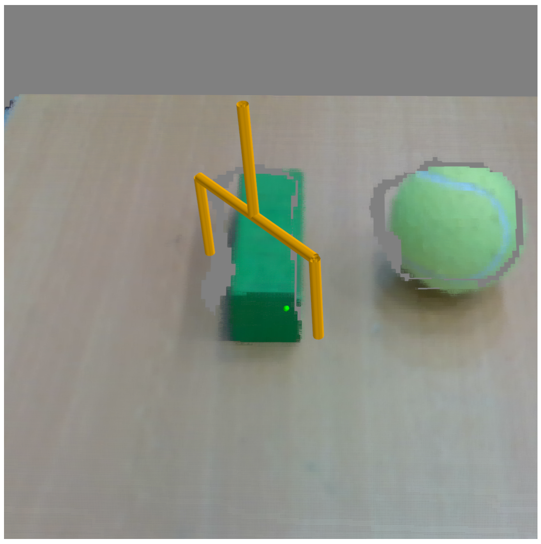
In our **140** grasp experiments conducted on the robot, **26** grasps were unsuccessful in being executed by the robotic arm. The majority of these failures occurred during the grasping of objects with complex shapes, and the primary reasons for failure were attributed to Limitation 1 and Limitation 2, as discussed above.

5.2 Multi-Object unguided grasping

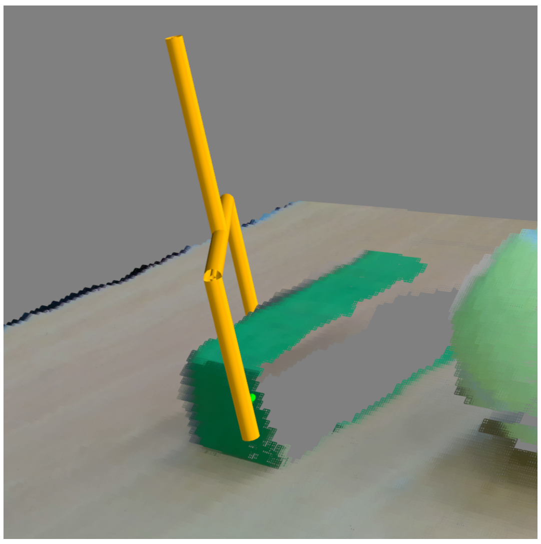
In the multi-object unguided grasping experiments, a total of five attempts were conducted, with the criterion for completion being whether all objects in the scene were successfully



(a) Scene in Camera View



(b) Predicted grasp



(c) Side view

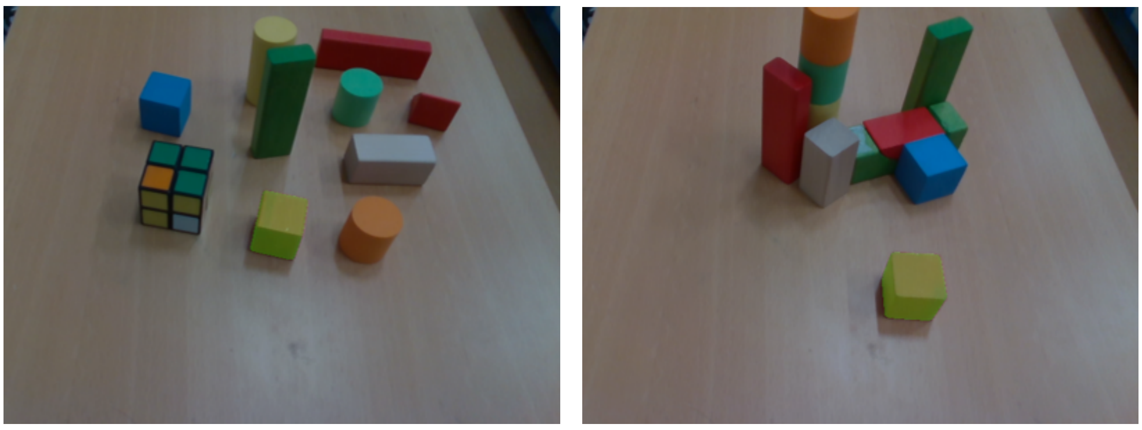
Figure 5.5: Case where no suitable grasps are generated due to Point Cloud

grasped (i.e., whether the scene was cleared). Among the **5** scenarios, **2** were not successfully completed. Here, only the results of these two instances are presented and discussed. It is noteworthy that the prompt of the detection model (Grounding DINO) was given as "object" solely to filter out the background.

The images corresponding to these two scenarios are depicted in the left and right panels of Figure 5.6, while their corresponding sequentially generated grasping processes are delineated in Figures 5.7 and 5.8. It is noteworthy that the final step highlighted with a red border in both Figure 5.7 and 5.8 represents the specific step where the grasping attempt failed.

Based on the experimental results on this topic, one advantage and one limitation of the pipeline can be summarized as follows:

- **Advantage**: The grasping model consistently succeeds in generating collision-free grasps in multi-object scenes, while the detection model and the segmentation model reliably separate graspable objects from the background.



(a) Scene 1 **(b)** Scene 2

Figure 5.6: Multi-object picking experiment

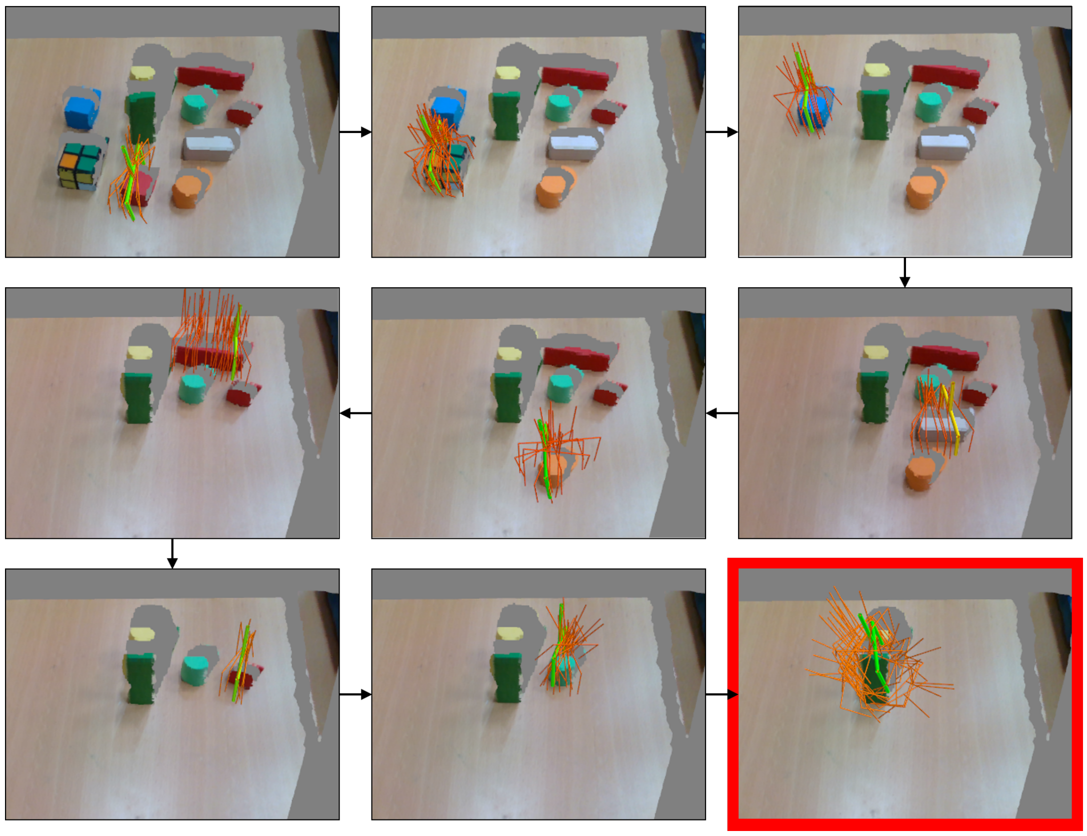


Figure 5.7: The sequential grasping of objects of Scene 1

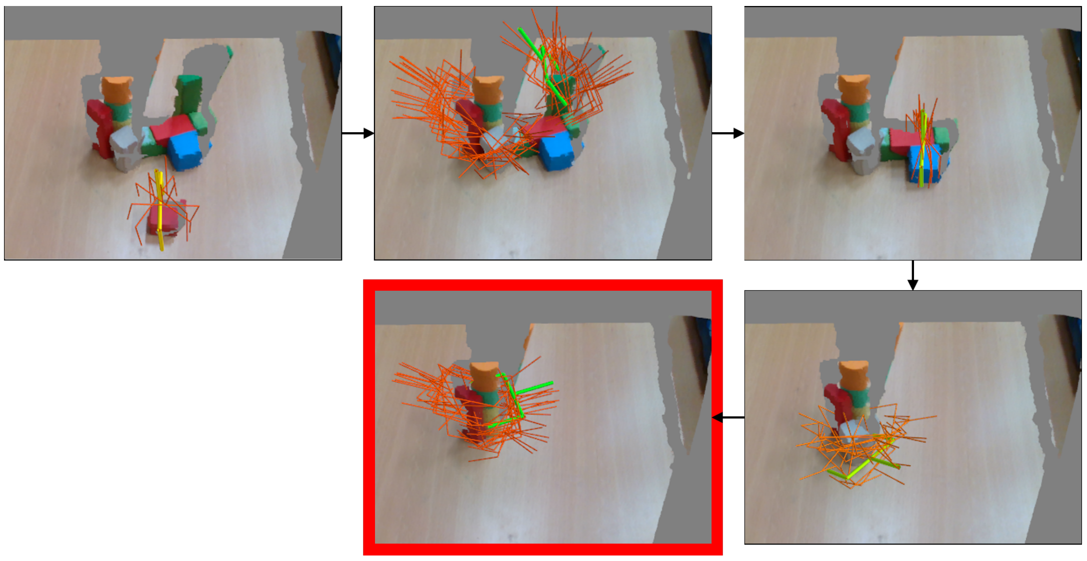


Figure 5.8: The sequential grasping of objects of Scene 2

- **Limitation**: The grasping model tends to treat closely spaced or physically adjacent objects as a single entity for grasp generation. This tendency is evident in the failure steps in Figure 5.7 and in almost every step in Figure 5.8. This characteristic primarily arises because grasping is based on the $2.5D$ point cloud of the entire scene, making it difficult to distinguish stacked objects without color information, particularly when the shapes of the objects physically coincide, as illustrated in the example in Figure 5.7.

5.3 Multi-object guided grasping

This experiment investigates object detection using the Grounded SAM model (Grounding DINO + Segment Anything). But we mainly evaluate Grounded DINO in this experiment rather than Grounding SAM given that the former is the part of Grounding SAM concerned with object detection. By placing multiple objects on the table the objective is to identify the correct object and assign it the highest confidence score out of the detected objects. We evaluate how sensitive the model is to changes in the input prompt, objects in the scenes, or the background.

Number of Trials	Fail on Detection	Fail on Grasping Estimation	Success
50	8	6	36

Table 5.2: Result of Multi-object guided grasping

50 experiments were carried out with guided grasping tasks, some examples are shown in Figure 5.9, 5.10, 5.11, 5.12, 5.13, 5.14 and 5.15. In the experiments, we found that Grounding DINO performs well in directional object recognition tasks when provided with precise object names. When combined with SAM and Contact GraspNet, the pipeline can effectively accomplish directional object-grasping tasks, as evidenced by the grasps generated in Figures 5.9 to **4**. However, we also noted that the results generated by the Grounding DINO model when faced with descriptive prompts are not always reliable. Figures 5.13 to 5.15 depict

several examples of directional prompts, which were successful in these cases, yet in actual experiments, attempts involving directional prompts have proven to be relatively unsuccessful.

Furthermore, the Grounding DINO model also suffers from the typical low interpretability issue inherent in visual-language deep learning models. In our experiments, we encountered instances where the model could detect screws but not screwdrivers, or failed to recognize a broken cup placed separately; however, when another intact cup was present in the scene, the model identified the broken cup as a cup. The reasons behind these occurrences remain unclear.

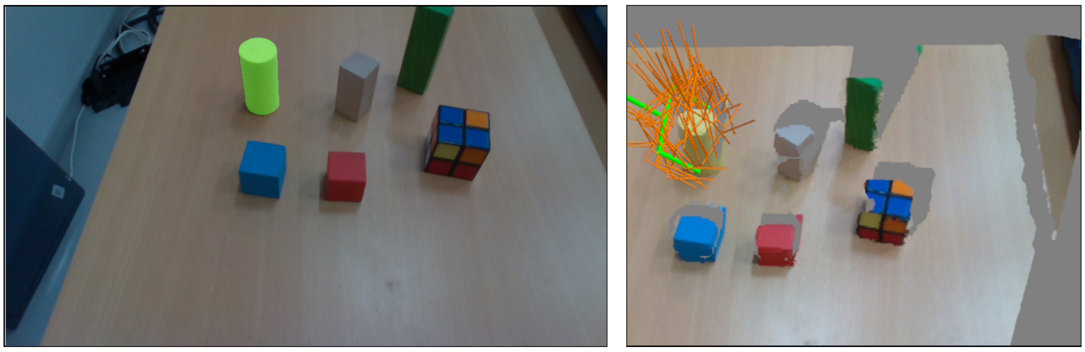


Figure 5.9: Prompt:"The Yellow Cylinder"

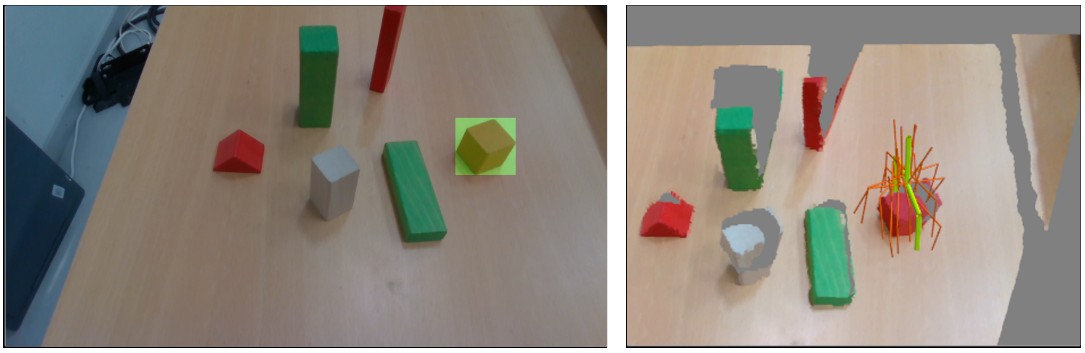


Figure 5.10: Prompt:"The Red Block"

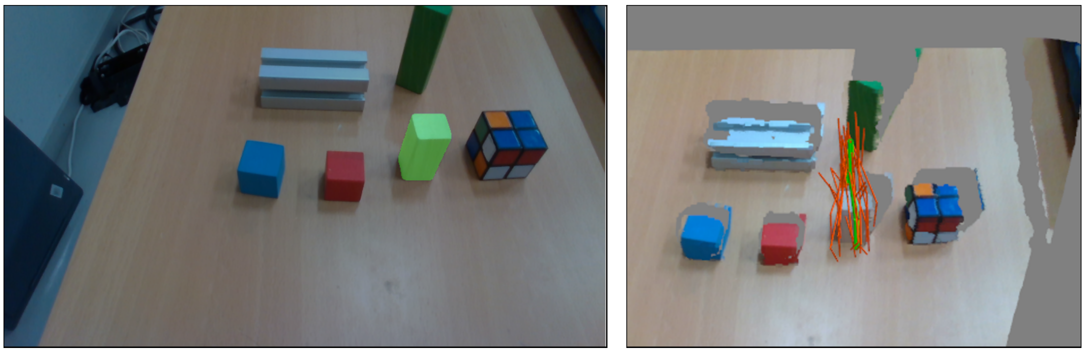


Figure 5.11: Prompt:"The Wooden Block"

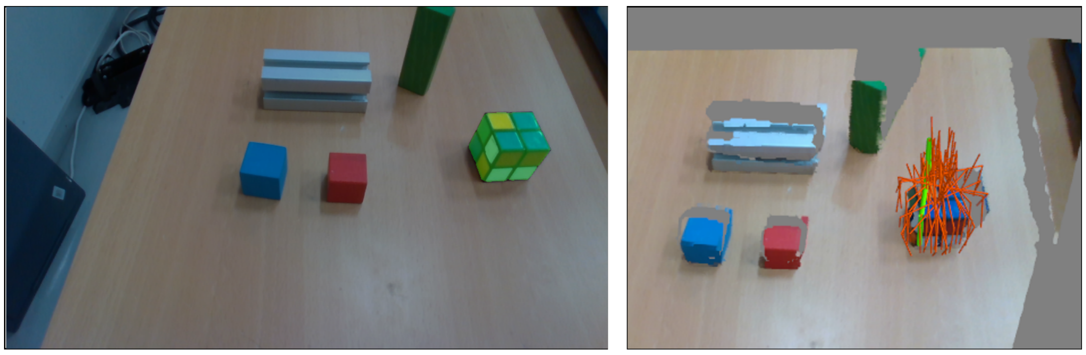


Figure 5.12: Prompt:"The Rubik's cube"

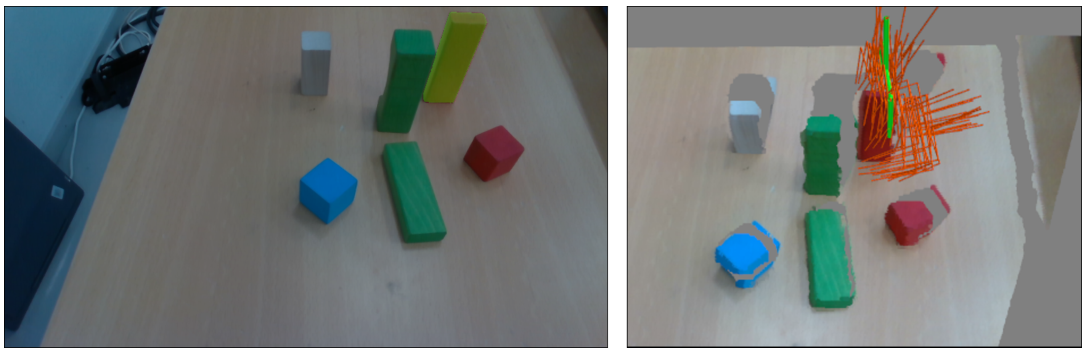


Figure 5.13: Prompt:"The Object on the back"

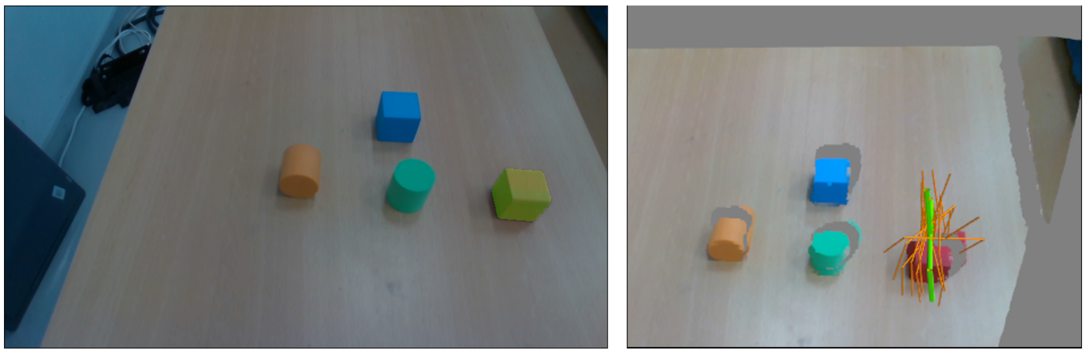


Figure 5.14: Prompt:"The object on the right"

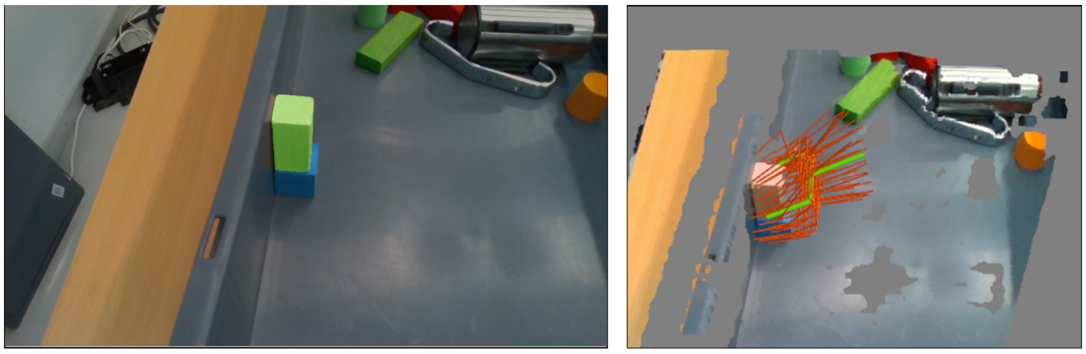


Figure 5.15: Prompt:"The object on the top"

Part III

Implementation on Tetra-Pak Box Picking

Chapter 6

Tetra Pak Box Picking

The Tetra Pak box that this project specified to pick is shaped as Figures 6.1, with fake labels that shown in Figure 6.2.

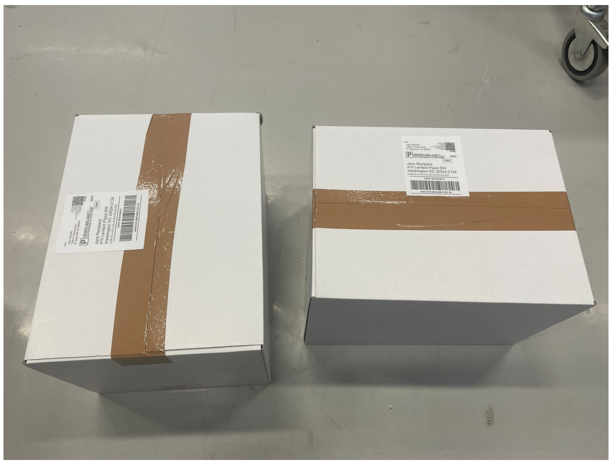
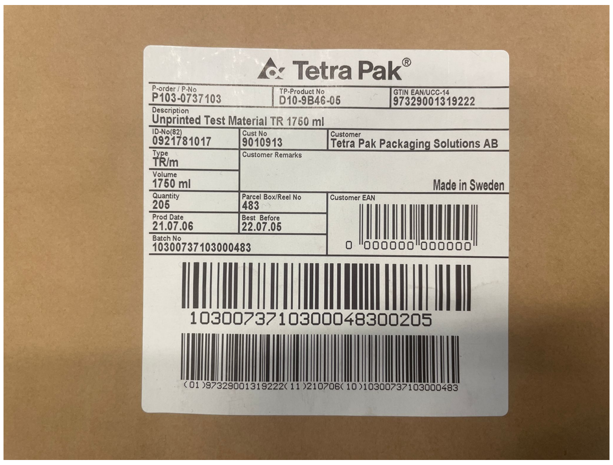


Figure 6.1: Tetra Pak Box to pick



Tetra Pak®

P-order / P-No
P103-0737103

TP-Product No
D10-9B46-05

GTIN EAN/UCC-14
97329001319222

Description
Unprinted Test Material TR 1750 ml

ID-No(82)
0921781017

Cust No
9010913

Customer
Tetra Pak Packaging Solutions AB

Type
TR/m

Customer Remarks

Volume
1750 ml

Made in Sweden

Quantity
205

Parcel Box/Reel No
483

Customer EAN

Prod Date
21.07.06

Best Before
22.07.05

Batch No
10300737103000483

0 000000 000000

10300737103000048300205

(01)97329001319222(11)210706(10)10300737103000483

Figure 6.2: The label on the box

Again, the purpose of the new pipeline is to adopt the existing methods on the specific Tetra Pak box and make the process more robust. As discussed in section 5.1 of Chapter 5, the Contact-Graspnet is not good at generating picking poses for objects that the depth camera can not fully capture, as both boxes are too big for the camera to capture the depth information of the full body, this model can not really perform well Tetra Pak Boxes. Further, given the size and the weight of Tetra Pak boxes compared to the normal object of the previous part, it is believed that such tasks are better handed over to the gripper with suction cups, as shown in Figure 6.3.

With the change of the gripper, also considering this picking task is to a specific industrial

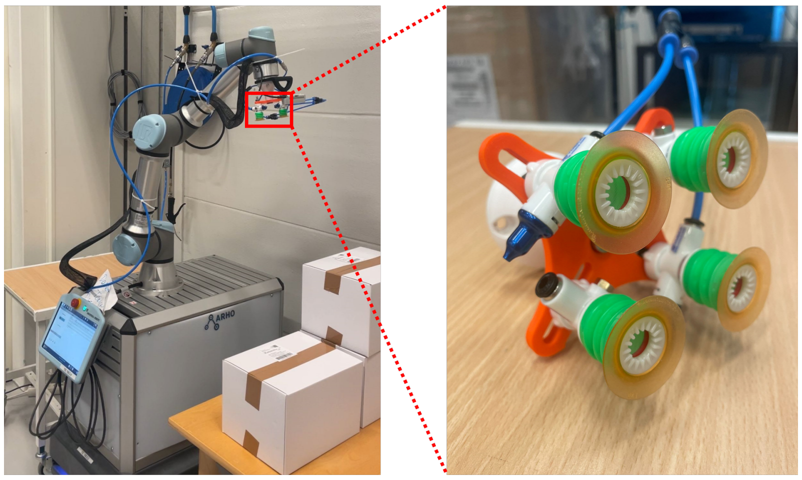


Figure 6.3: Heron Robot and Suction Cup

product, there is a need to propose new methods for generating picking poses and to make the results of visual inspection more robust, making the entire grasping process more robust. The new pipeline, as shown on Figure 1.2, will be explained in the next section.

6.1 Theory

Due to the size of the boxes, it is found that the depth image usually can only capture the upper plane and a little bit of the side shape of the box, making it difficult to estimate a picking pose out of the whole box. However, given that now the gripper is switched to the suction cup, and the Tetra Pak box can be seen as a box with evenly distributed weight, the box can be picked by the suction from the center point of the upper surface plane with the gripper standing top-down straight, therefore the center point of the plane is all that is needed to pick the box. But to do this, the first step is actually find where the boxes are located in the scene.

6.1.1 Tetra Pak Box Detection

The Open-Vocabulary Detection and Segmentation Model that this project adopted, Grounded SAM, is powerful and flexible for detecting unknown objects but does not necessarily give consistent results with the same prompt. To detect the Tetra Pak, the initial usage of the prompt was "Box with label", this gives all the boxes with label in one single bounding box (as Figure 6.4), which is not ideal as the separate boxes remain hard to find.

After several attempts, it is found that the most suitable prompt to use to detect Tetra Pak boxes is "box.label" with both "box" and "label" passed in as in name of classes, in other words, this prompt asks the model to detect objects that can be considered as boxes and objects that can be considered as labels at the same time, given well clear and well-separated detection results as in Figure (6.5).

It is interesting that with the "box with label" along, none of the labels were detected, and individual boxes were not indicated, while when "box" and "label" were detected together but

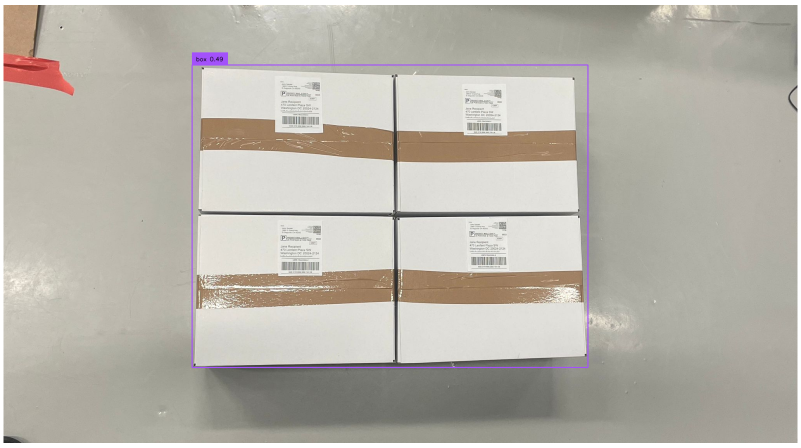


Figure 6.4: Detection result with the prompt "Box with Label"

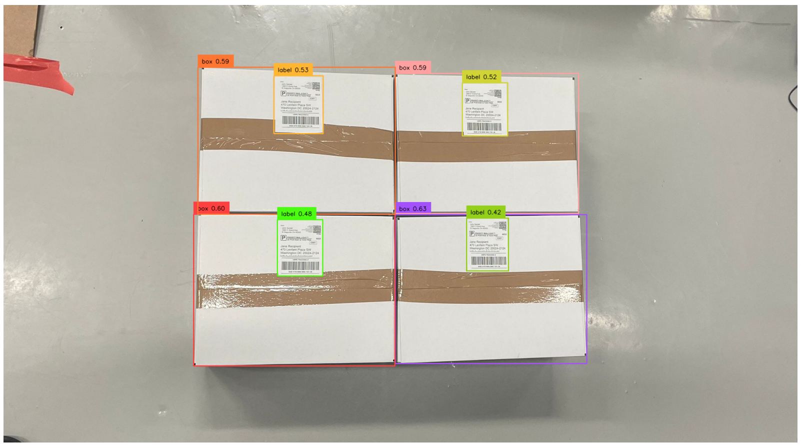


Figure 6.5: Detection result with the prompt "Box.Label"

separately, both were marked clearly. The reason for this phenomenon might be due to two possible hypotheses:

1. The model has a sort of associative ability, making it easier to identify the boxes after the labels were detected as these two classes can be related.

2. "labels" are more recognizable than "boxes" for the model.

Proving the first hypothesis is not straightforward, but this is a way to show if the second one somewhat makes sense. One way of showing this is to compare the "confidence" of bounding boxes. The confidence values (also called logits) are the values on the upper left corner of each bounding box, showing how confident the model thinks this is a "box" or "label". By using the prompt "box . label" on 3000 images that contain 4 true Tetra Pak boxes and corresponding labels, like Figure 6.5 (meaning ground truth is known), and taking the confidence of the bounding boxes of the detected objects, the joint histogram of the confidence of "box" and "label" can be shown as Figure (6.6)

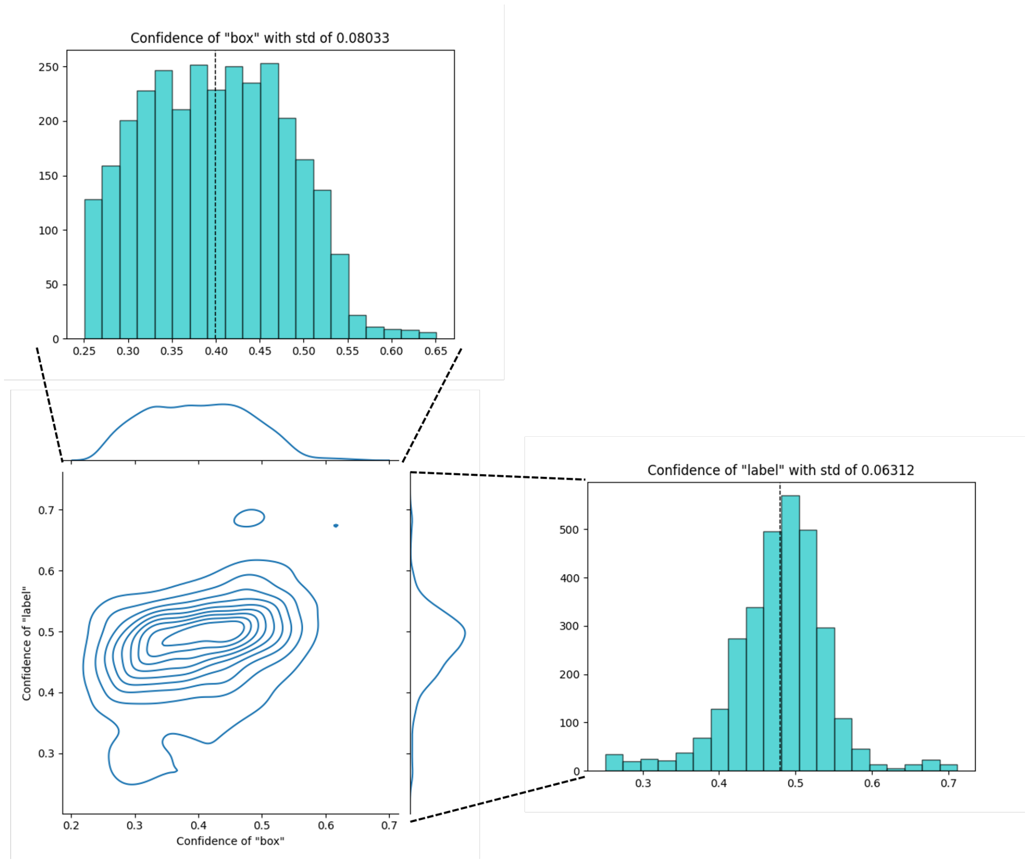


Figure 6.6: Joint histogram of two classes

This joint history tells the histogram of "label" comes with a lower standard deviation and higher mean compared to the histogram of "box", also, the peak of the histogram of "label" is higher than the peak of the other histogram, all of these indicating that the model tends to have higher confidence in detecting labels, which in turns helps in the detection of the boxes, thanks to the associative ability of the detection model.

6.1.2 Plane Estimation

Once the boxes are detected, estimating the normal, and center of the upper plane is the next step. Given the input as a point cloud of (x, y, z) points, a plane can be estimated by using a plane equation combined with RANSAC, as described in algorithm 1.

Algorithm 1: Plane Estimation with RANSAC

```
$pts, th, mi \leftarrow$ points, threshold, maximum iteration;
$N \leftarrow 1$;
$optimal\_norm, optimal\_inlier \leftarrow (0, 0, 0), 0$;
while $N \le mi$ do
    $pts\_sample \leftarrow$ Random($pts$, 3) % Randomly sample 3 points ;
    $norm \leftarrow$ Plane($pts\_sample$) % Find the plane that is defined by three points
      and get the normal ;
    $count \leftarrow 0$;
    for $pt \in pts$ do
        $l \leftarrow$ Distance($pt, norm$) % Compute the distance from point to plane;
        if $l \le th$ then
            $count \leftarrow count + 1$;
        end
    end
    if $count > optimal\_inlier$ then
        $optimal\_inlier \leftarrow count$;
        $optimal\_norm \leftarrow norm$;
    end
end
return $optimal\_norm$
```

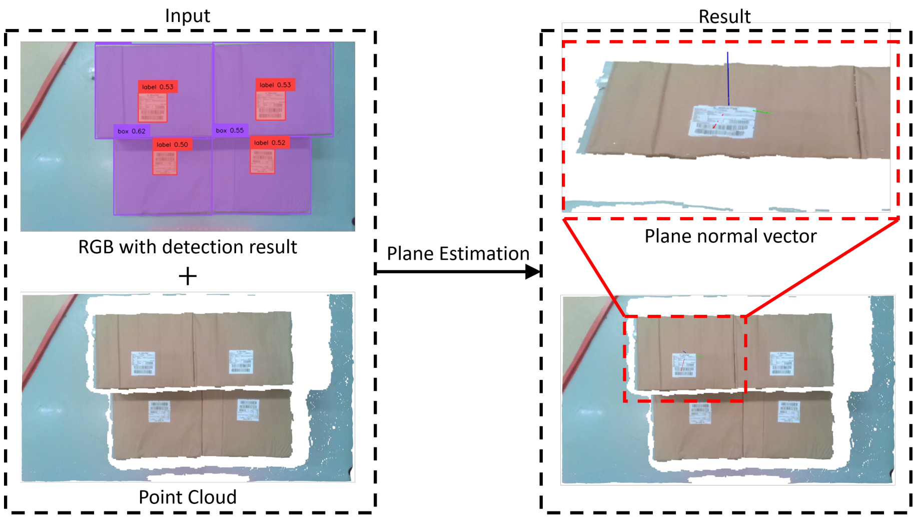


Figure 6.7: Plane estimation for boxes

The plane estimation is done on the point cloud that is cropped by the segmentation mask within the detection result as described last section. This process can be visualized as Figure 6.7.

Once the plane points are found, the center of the plane can then be found by taking the mean of the points.

6.2 Result

To evaluate this new picking method, 100 trials were carried, and the result is as table 6.1

Number of Picks	Fail on Detection	Fail on Plane Estimation	Success
100	17	4	79

Table 6.1: Result

It is found that the main obstacle to achieving good robustness is the detection system, it is easy to detect something else in the scene as a Tetra Pak box. Other than that, our pipeline gets fine results for a vanilla method.

Chapter 7
Conclusion

We can summarize our findings in the following aspects:

- For the Grasping generation model:
  - Except for the discussed edge cases, the Contact-GraspNet model performs very well as it did not fail in any single object-picking test.
  - The partial point cloud presents a big limitation due to the missing point on the invisible views to the camera. Sometimes these invisible views include the only contact points that can generate a suitable grasp. Enhancing the partial point cloud to reconstruct and fill in those points as a preprocessing step could be a viable solution.
  - The scoring of the grasps does not take into account the weight distribution of the object which could lead to weird grasps. One possible solution is to consider how far the grasp is to the center of mass of the object.
- For the Grounded-Segment-Anything model
  - Using only the segmentation as a filter in a cluttered scene does not guarantee grasping only the required object.
  - While this open-vocabulary model works very well with common objects, it struggles with consistency. Variations in prompts, scenes, and background often yield different results.
  - The lack of consistency makes it challenging to apply tasks that require high robustness, though certain levels of certainty can still be guartened when used with designed methods.

Overall, our pipeline demonstrates fine performance for both normal object picking and Tetra Pak box picking. The limitations existing in the current models can serve as directions for further improvements in the pipeline's development in the future.

References

[1] Richard Bronson, Gabriel B Costa, John T Saccoman, and Daniel Gross. Linear algebra: Algorithms, applications, and techniques. 2023.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[4] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. ACRONYM: A large-scale grasp dataset based on simulation. In *Under Review at ICRA 2021*, 2020.

[5] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[7] Daichi Kawakami, Ryoichi Ishikawa, Menandro Roxas, Yoshihiro Sato, and Takeshi Oishi. Learning 6dof grasping using reward-consistent demonstration. *arXiv preprint arXiv:2103.12321*, 2021.

[8] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[10] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M. Ni, and Lei Zhang. Dn-detr: Accelerate detr training by introducing query denoising, 2022.

[11] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10975, 2022.

[12] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. Dab-detr: Dynamic anchor boxes are better queries for detr, 2022.

[13] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[14] Jens Lundell, Francesco Verdoja, and Ville Kyrki. Ddgc: Generative deep dexterous grasping in clutter. *IEEE Robotics and Automation Letters*, 6(4):6899–6906, 2021.

[15] Jens Lundell, Francesco Verdoja, Tran Nguyen Le, Arsalan Mousavian, Dieter Fox, and Ville Kyrki. Constrained generative sampling of 6-dof grasps. *arXiv preprint arXiv:2302.10745*, 2023.

[16] Matthias Mayr, Francesco Rovida, and Volker Krueger. Skiros2: A skill-based robot control platform for ros. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6273–6280. IEEE, 2023.

[17] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2901–2910, 2019.

[18] Rhys Newbury, Morris Gu, Lachlan Chumbley, Arsalan Mousavian, Clemens Eppner, Jürgen Leitner, Jeannette Bohg, Antonio Morales, Tamim Asfour, Danica Kragic, et al. Deep learning approaches to grasp synthesis: A review. *IEEE Transactions on Robotics*, 2023.

[19] Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion*. *Acta Numerica*, 26:305–364, 2017.

[20] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[21] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[23] Joseph Redmon and Anelia Angelova. Real-time grasp detection using convolutional neural networks. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 1316–1322. IEEE, 2015.

[24] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[26] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13438–13444. IEEE, 2021.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[28] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.

[29] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5745–5753, 2019.