UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN
COMPUTER ENGINEERING

"From Perception to Grasp: a Real-Time Framework for Object-Agnostic Human-to-Robot Handovers"

Relatore: Prof. Stefano Ghidoni

Laureando: Leonardo Parancola

Correlatori: Prof. Matteo Terreran
Dott. Leonardo Barcellona

ANNO ACCADEMICO 2023 – 2024

Data di laurea 10/07/2024

Abstract

The handover of objects consists of a joint action of a giver transferring an object to a receiver. Providing such capability to collaborative robots is crucial for human-robot collaboration in manufacturing scenarios, especially when people are involved. The objective of this work is to provide a general framework for a real-time, object-agnostic human-to-robot handover. After the generation of a point cloud of the scene by means of a single RGB-D sensor mounted on the robot's gripper, the proposed method comprises three phases. The first one involves the segmentation of the scene, which allows to distinguish between the human hand and the object being held. The second phase consists of generating a set of possible grasps from the elaborated scene. Finally, the third phase regards the selection of the feasible grasp poses according to three defined criteria. The proposed approach consists of constructing a possible implementation for the Human-To-Robot handover task, leveraging and modifying models found in the literature. In particular, a faster version of the EgoHOS [1] model, *Fast-EgoHOS*, for the segmentation task has been proposed and compared to its original implementation, named *Complete-EgoHOS*. Moreover, a comparison has been conducted between three state-of-the-art models for grasp generation, which are GraspNet [2], 6-DoF GraspNet [3], and Contact-GraspNet [4]. Based on the best segmentation-grasp detection model configuration, identified through an offline evaluation in terms of accuracy, execution time, and grasp quality, an online evaluation is performed. This is conducted by measuring the success rate and the time needed to perform an handover attempt. The presented approach allows us to achieve a value of IoU of *Fast-EgoHOS* of 78.8% compared to the 82.84% of *Complete-EgoHOS*, with a significant advantage in terms of inference time. In the online evaluation, a grasp success rate of 82.9% is achieved with the *Fast-EgoHOS-Contact-Graspnet* configuration and of 80.3% with *Fast-EgoHOS-Graspnet*. Results are obtained considering a set of 19 distinct objects presented in various positions.

Contents

Chapter 1

Introduction

In recent years, there has been a growing interest in the collaboration between humans and robots in industrial and social environments, especially in dynamic and unstructured settings. This is also evidenced by the increasing number of global sales and integration of collaborative robots in industry [5], [6], which are designed to work alongside human workers. One of the key challenges in developing *cobots* is the development of appropriate systems to effectively respond and perceive to external stimuli, especially in workspaces that may include the presence of people [7]. These robots are employed in various scenarios, such as warehouse management [6], healthcare [8] and agriculture [9], where flexibility, dexterity, and adaptability are important factors to be considered.

The interaction between people and collaborative robots can be summarized as *cooperation* and *collaboration*. These terms are typically used interchangeably in Human-Robot Interaction (HRI) research, although they are not the same concept. The cooperation between agents is described as a sequence of independently executed actions towards a shared goal. In contrast, collaboration is described as a sequence of shared actions towards a common goal [6], [10]. Collaboration between humans and robots can be exemplified as the combined actions needed to achieve a common task, i.e. assembling an object. In this particular context, a robot may collaborate with a human operator by passing the appropriate object at the right time, or, in an industrial environment, helping him to transport part of it.

The ability of robots to interact with diverse and unpredictable environments is an essential skill for human-robot interaction, with object manipulation being an important part of it [11]–[15]. One of the ways robots can interact with objects is by grasping them, which is the action of constraining an object by applying forces and torques on its surface [16].

For a robotic arm, manipulating the environment can be challenging in terms of perception, planning and control. Let us take as an example the pick-and-place problem, which requires the robot to move an object from one location to another. The perceptual aspect of this problem

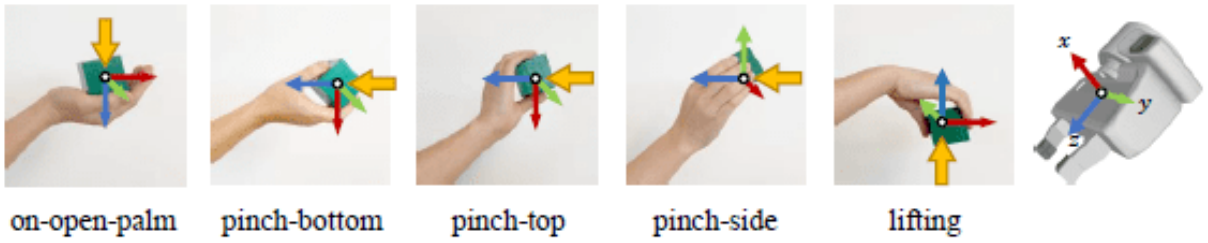


Figure 1.1: Illustration taken from [19]. This image shows an example of how grasps in the H2R paradigm can be pre-planned according to the manner in which an object is held by a person.

concerns the robot's ability to perceive its surrounding environment, identifying the objects in the scene. In particular, the perception must focus on the object to be picked up and on the contact points that would lead to a robust grip. Once the target object has been selected, a set of possible grasps must be computed, selecting only those which are located within the robot's workspace. At this point, the planning and control systems need to identify which is the optimal route to reach the generated grasp, while avoiding other elements present in the scene. The overall task can be computationally expensive, proportional to the number of robot degrees of freedom to be managed [17]. The task can also be challenging depending on the system's knowledge about its surrounding, particularly in the case of objects with unknown characteristics, such as size, weight, or shape.

The pick-and-place task can be further complicated when a human being is involved, for instance, when an object needs to be picked up from their hand or vice versa. In such cases, the action performed is termed *object handover*, which is the act of transferring an object from a giver to a receiver [15]. The challenges are mainly the same as the previous example, with a particular emphasis on the safety aspect. In fact, the robot must take into account the motions and the unpredictability of the person.

According to the definition of object handover, there are three distinct paradigms between human and robot agents: human-to-human (H2H), robot-to-human (R2H), and human-to-robot (H2R). H2H interactions, in the context of handover, refer to the joint actions between two humans exchanging an object. Although this type of actions may appear natural to us, it involves different and complex aspects, from the representation of the goals to the processes for accomplishing them. These processes may include monitoring the execution of the task and predicting the action the other agent may perform. People tend to form representations either of their own goals and tasks and those of their partner. They also monitor the advancement of the tasks and try to predict the outcome of one's actions in the immediate future [18].

On the other hand, R2H and H2R handover paradigms involve the presence of a robot as the giver or the receiver. In both cases, the main actions to be performed are similar to the ones of the H2H paradigm. To better clarify this concept, it is possible to define a sequence of actions

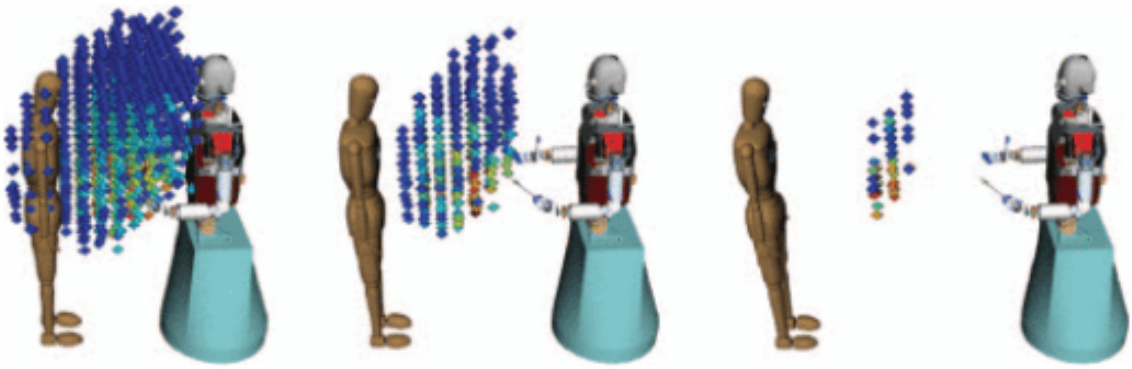


Figure 1.2: Illustration taken from [20]. This image shows where a suitable location for human-robot interaction can be located. In particular, it can be located at the intersection between both human and robot workspace. The image illustrates this area for various distances (70cm, 110cm, 150cm) between the two agents.

that are in common to the three paradigms. The first one involves the communication of intents, which is the ability of communicate and comprehend the intentions of the other agent. In the case in which the human agent passes an object to the robot, the intent of the person can be interpreted by robots, for instance, analysing human's kinematic features [21]. The second action is the grasp planning, which consists of the giver planning their motions considering the task of the receiver. For instance, in case the task involves a handover, the giver considers how to grasp the object so as to offer it to the receiver in the best way possible. An example of this is provided by Figure 1.1, where some predefined grasps poses are defined. The third action is about the perception of the scene, which is the ability of gathering information about the surrounding environment. For human-robot handovers, it consists of distinguishing the human's body, the object to pick and the elements being part of the environment. The fourth one regards the location in which the handover should occur, which must be reachable by both agents. In H2H handovers, it has been observed that the location occurs midway [22], while for the H2R and R2H paradigms, it should take place at the intersection between human and robot workspaces, as depicted in Figure 1.2. In particular, the H2R handover generates a set of feasible grasps starting from the knowledge provided by the perception system. Some examples based on deep learning techniques, which will be discussed in Chapter 2, include GraspNet [2], 6-DoF GraspNet [3], Contact-GraspNet [4], or AnyGrasp [23]. Finally, the fifth action is the motion planning and control, which consists of the actuation of the motion for performing the task. In the H2H interaction, movements are generally smooth [24]. In contrast, for the other paradigms, motions are represented as a sequence of separate and successive phases. These involve the motion legibility and predictability, which are the ability of one agent to understand and predict the other agent's movements. Furthermore, the phases should also allow for the robot's motion robustness, reac-

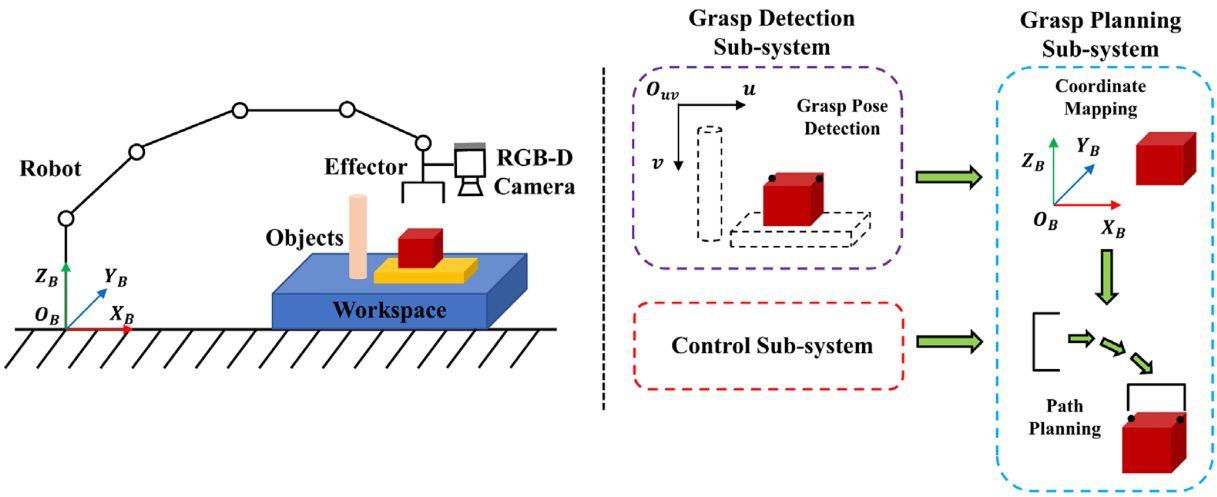


Figure 1.3: Illustration taken from [12]. Left: A typical robotic arm configuration with an eye-in-hand camera. This is the same configuration that will be used in the experimental part. Right: The three aspects that are involved in object handover, namely, grasp pose detection, to generate a set of suitable grasps for the object, grasp planning, to generate a trajectory, in order to successfully grasp the object, and the control subsystem, to actually move the robotic arm.

tivity and context awareness, which consists of the adaptation of the motion to changes in the environment [25].

This sequence of actions, in the context of handover, can be divided into two main phases [15]: the pre-handover phase, which involves communication between the two agents about their intentions and the grasp detection, and the physical handover, which begins when the receiver makes contact with the object and ends when the object is fully in the hold of the receiver.

Although R2H and H2R interaction paradigms involve the presence of a robot as an agent, they face different issues, especially in terms of safety [14], [15].

Some of the challenges that R2H handovers may face are: the human comfort, where the robot performs the task according to human expectations, such as where, how quickly, and in which direction objects should be handed; the proactivity of the robot, in which it needs to decide the timing and method of the handover; and the determination of when to release the object.

Instead, H2R handovers have different priorities, such as: the uncertainty of human behaviour, since humans may behave differently when passing an objects, e.g., in terms of speed and grasp location and orientation; a real-time response to the human motion, which requires efficient sensor data and fast action generation, e.g., to cope with the hand movements and obstacles; and a safe grasping strategy, in which the robot must plan collision-free trajectories and grasps pick up the object safely, without pinching the human operator or colliding with his body part.

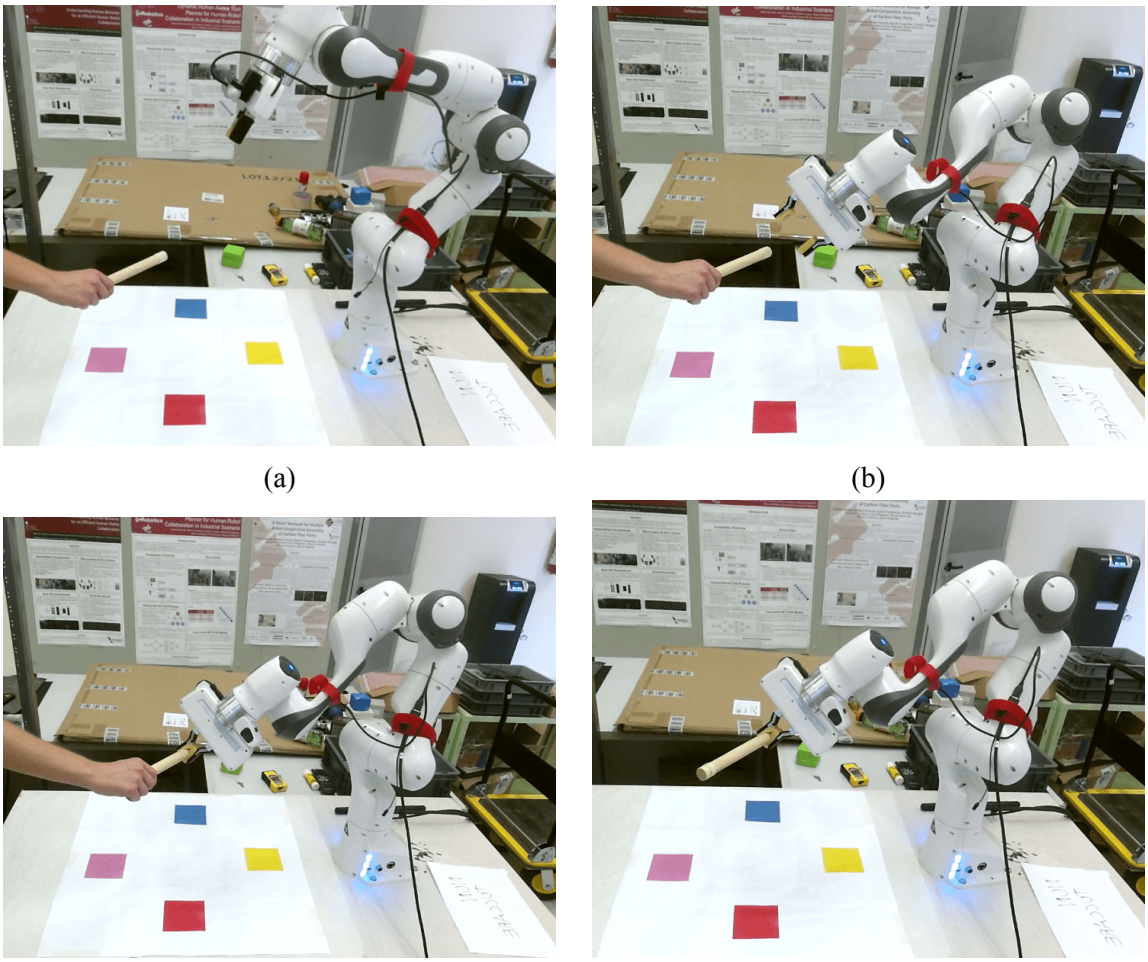
A complete robotic grasp system usually comprises three main modules [12]: grasp pose detection, grasp planning and control subsystem. These are depicted in Figure 1.3. The grasp

pose detection part is responsible for generating a dense set of grasp poses on the target object's surface, eventually with the object's pose in space. The grasp planning subsystem has the role of representing the grasps in the robot's coordinate frame and generating a feasible path to guide the manipulator to the target pose. Finally, the control subsystem module is responsible for determining the inverse kinematics solution to physically move the robot.

The first module requires the implementation of a perception system. To achieve this objective, one or multiple cameras need to be installed to provide a comprehensive view of the scene. According to their number and placement, three system configurations can be identified: the *fixed configuration*, or *eye-to-hand*, in which a single camera is mounted on a fixed location; the *mobile configuration*, or *eye-in-hand*, where a single camera is attached to the robot; and the *hybrid configuration*, in which multiple cameras are involved [17]. The role of the perception system is to enable the robot to acquire knowledge of its surrounding environment. In the context of object handover tasks, a visual system may comprise an RGB-D camera, that is, a camera with an incorporated depth sensor, in a fixed or mobile configuration. The use of this type of sensor is to facilitate the elaboration of the scene, providing depth information that would otherwise be unavailable to a single RGB camera. Indeed, as it will be discussed in Chapter 2, depth data can be utilized as input for image processing algorithms or to assist the robot in grasping the target object [19], [26]–[28]. Moreover, given the depth information, it is possible to represent the scene in the form of point clouds, which are a representation of three-dimensional data. A point cloud is a collection of data points, defined by their x, y, and z coordinates, in a 3D space. Figure 1.3 illustrates a typical robotic arm configuration for the pick-and-place task, equipped with an *eye-in-hand* RGB-D camera. This is the same configuration that will be used in this thesis, which is possible to see in Figure 1.4.

A possible way of processing an input image or point cloud is through the use of segmentation techniques, which consist of partitioning the input into distinct regions according to their characteristics. Segmentation techniques are referred to as 2D or 3D segmentation, respectively, depending on the type of input data, which may be 2D images or 3D point clouds. In an object handover task, in which the involved parts are a human and a robot, the visual system, together with the segmentation techniques, can be utilized for distinguishing the objects from the human body. Techniques for segmenting the human hand from the interacting object, such as Ego-HOS [1], have already been developed. However, these are not always suitable for real-time applications, due to their low inference speed.

Furthermore, segmentation masks can be used to remove certain of the generated grasps that may result in the robot pinching or hitting the human body. Grasps can be generated directly on the image plane [29], [30] or on the point cloud of the object [2]–[4], [23], [31]–[33], based on the available data. Consequently, they can be visually be represented in the image plane in



(a) (b)

(c) (d)

Figure 1.4: The figure illustrates an example of object handover during the experimental phase. Figure 1.4a shows the robot analysing the scene to segment the hand from the object and compute a set of feasible grasp poses. In Figure 1.4b the robot moves to an approach position. Figure 1.4c and Figure 1.4d display the robot grasping the object and keep it.

the form of rectangles, which reflect the physical characteristics of the gripper, or in the form transformation matrices, which represent the translation and rotation the end effector frame has to perform in order to be aligned to the computed grasp pose.

The thesis will be based on the H2R handover task, which can be visualized in Figure 1.4, that illustrates an handover attempt of the implemented approach using a wooden stick. In particular, it focuses on the segmentation part, which is responsible for distinguishing the hand from the object, and the grasp detection component, which is responsible for generating a set of feasible grasps. The main contributions are the following:

1. A faster method, called *Fast-EgoHOS*, for the segmentation of hand and object, which is designed to achieve good accuracy while maintaining the system fast, utilizing a smaller network with emphasis on a post-processing phase;

2. The adaptation of three different state-of-the-art grasp generation models, namely GraspNet [2], Contact-GraspNet [4], and 6-DoF GraspNet [3], originally designed for objects placed on a plane surface, to work in the H2R context;

3. An offline evaluation of either the segmentation and grasp generation modules is conducted in terms of execution time, accuracy, and ability to generate feasible grasps, with the objective to select the best segmentation-grasp generation pair;

4. An online evaluation, of the selected configuration, is performed in terms of success rate and execution time on a set of 15 distinct objects, varying in size and shape, for a total of 456 handover trials. This is followed by an analysis of the encountered problems and limitations.

This thesis is organized as follows. Chapter 2 offers a review of the state-of-the-art technologies regarding 2D and 3D segmentation and grasp generation, along with some examples of complete handover pipelines found in the literature. Chapter 3 presents the proposed solution for this problem, while Chapter 4 describes the obtained experimental results together with the encountered difficulties and limitations.

Chapter 2

Literature Review

As discussed in [12], [14], a generic framework for the grasp detection system comprises three steps: grasp pose detection, grasp planning and the control subsystem. As previously stated in the introduction, the grasp pose detection necessitates the presence of a visual perception system, usually comprising RGB-D cameras to have both RGB data and depth information of the scene, along with the implementation of segmentation techniques, to distinguish the human body from the object to pick.

It is therefore possible to rearrange these steps into four in the following way, as illustrated in Figure 2.1. The first step concerns the visual perception system, which comprises the sensors that enable the robot to perceive the scene. Then, the second step consists of the scene understanding algorithms, which includes the visual algorithms that elaborate the scene, such as through segmentation. The third one is the grasp detection module, which consisting of the algorithms that allow the generation of a set of possible grasps on the object's surface. Finally, a motion planning module is required to physically move the robot to the selected grasp pose.

From the image, it can be observed that this type of framework is a closed loop. This is because, in the ideal case, a robot should be able to perceive the environment and react to any changes that occur within it. This could include the avoidance of obstacles or the motion of the target object to be picked.

This chapter is organized as follows. Sections 2.1 and 2.2 reviews the state-of-the-art technologies for the image and point cloud segmentation. Sections 2.3 revises the grasp detection

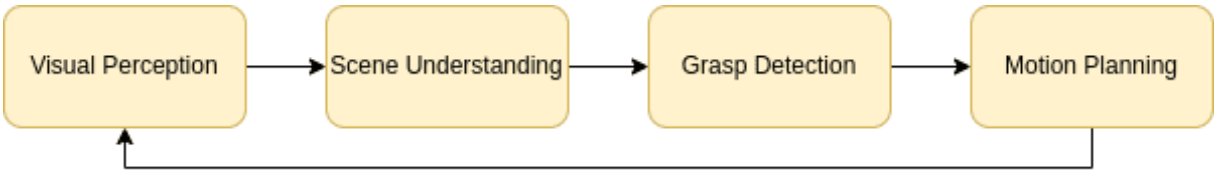


Figure 2.1: A human-robot object handover framework as described in [12].

taxonomy with the corresponding techniques, and Section 2.4 the used datasets and evaluation metrics. Finally, Section 2.5 briefly describes some complete pipeline frameworks for the H2R problem found in the literature.

2.1 Image Segmentation

Image processing refers to the manipulation and analysis of digital images, using computer algorithms. There are numerous ways in which an image can be elaborated and analyzed. One example is through feature extraction, which is defined as the process of extracting important and invariant characteristics of the image. Another is object detection, which is the process of recognizing which elements are present and where they are located in the image. Finally, image classification, which is the task of assigning a class of belonging to the image, according to its characteristics.

The application of machine learning to the field of computer vision is not a recent phenomenon, for example [34] applies it successfully in 1989. However, it has received more interest following the introduction of the AlexNet [35] model, which demonstrated the successful application of deep learning to the image classification task. Although this model reaches state-of-the-art performance with respect to other technologies, it suffers of some limitations, such as the high computational cost and overfitting. Moreover, deep networks may also suffer of the problem of vanishing or exploding gradients, especially in the context of Convolutional Neural Networks (CNNs) [36], an architecture for the image processing in machine learning. These two issues manifest during the learning process, resulting in either an insufficient growth of the neural network weight parameters, or alternatively, in an exponential growth.

More recently, the ResNet [37] architecture has emerged with the objective of overcoming some of the inherent limitations of deep networks. This approach has improved the efficiency in image-based tasks through the introduction of residual blocks, which address the issues of the vanishing and exploding gradients.

In the context of this thesis, the task of semantic segmentation is of particular relevance. It consists of the process of partitioning an input image into distinct regions and assigning to each pixel a specific category [38]. If there is the necessity to distinguish each of the regions as different instances of the same object, then the task is named instance segmentation. An example of instance segmentation is Figure 2.5, where left and right hand are distinguished as the interacting objects. Thanks to its ability of dealing with deep networks, the ResNet architecture is widely employed as backbone for image segmentation tasks [1], [26], [27], [29], [39].

A model that makes use of it for the instance segmentation task is EgoHOS [1]. The objective of this model is to infer a per-pixel segmentation mask of hands and objects from an input image,

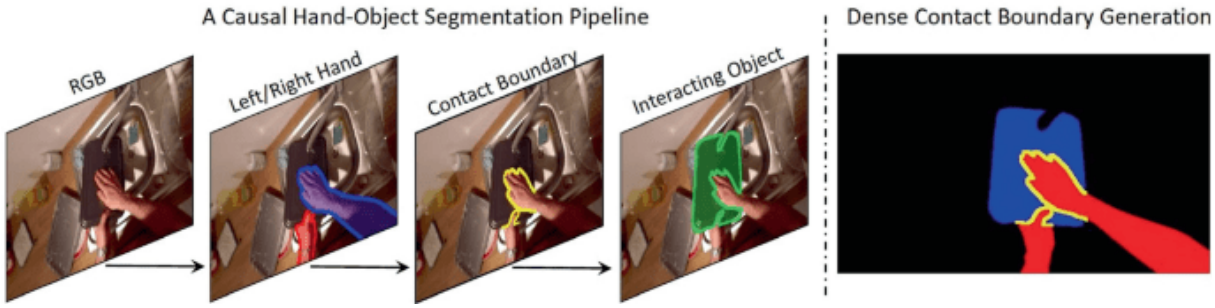


Figure 2.2: On the left, the three steps through which an input image passes. On the right, the resulting instance segmentation result is displayed. This result is provided by the EgoHOS model [1].

Figure 2.3

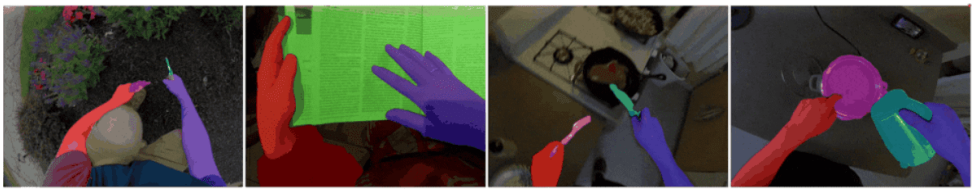


Figure 2.4

Figure 2.5: Example of the segmentation process applied on images. Example taken from EgoHOS [1].

as shown in Figure 2.2. In particular, the network is designed to segment the left and right hands with the corresponding interacting objects. The network then proceeds to elaborate the image in accordance with a three-step procedure. The initial stage consists of inferring the segmentation mask of left and right hands. The second step consists of inferring the contact area between the hand and the object, basing on the result of the previous step. Finally, the last phase consists of finding the mask of the interacting object with the knowledge provided by the previous step outputs.

2.2 Point Cloud Segmentation

The application of deep learning on 3D point cloud data is more recent than its application on RGB images. One of the earliest deep learning networks to accept this type of data is VoxNet [40], which demonstrated its efficacy in the object recognition task. In this work, 3D data was represented in the form of volumetric occupancy grid of size 32x32x32, which captures the probability of occupancy for each voxel. This representation suffers of the following limitations: the computational costs, especially for high-resolution point clouds; a limited scalability, since its performance is highly dependent on the data resolution; and the information loss, which is caused by the voxelization process. Other methods represent voxels data in a more space-efficient manner, by leveraging on octrees, a data structure used to organize 3D space. The resolution of octrees adapt recursively to the complexity of the subject, facilitating the representation of complex three-dimensional structures. This technique is employed by OctNet [41].

A deep learning architecture that significantly improved the performance on 3D data processing was PointNet [42]. The fundamental concept is to directly elaborate raw point clouds with a network's response that is invariant from the points permutations. The ability of this network to extract local and global features from point clouds demonstrates its efficacy for this type of data. Indeed, this network achieved state-of-the-art results in both classification and 3D data segmentation. An improvement of this architecture is PointNet++ [43], which recursively applies PointNet to the point cloud.

The PointNet and PointNet++ networks are currently employed in the contexts where 3D data, in the form of point clouds, is involved. As will be detailed in the next section, in the context of this thesis, 3D grasp detection methods make use of these architectures, for example, to evaluate the generated grasp poses [3], [33] or infer the approaching vectors on the point cloud [2].

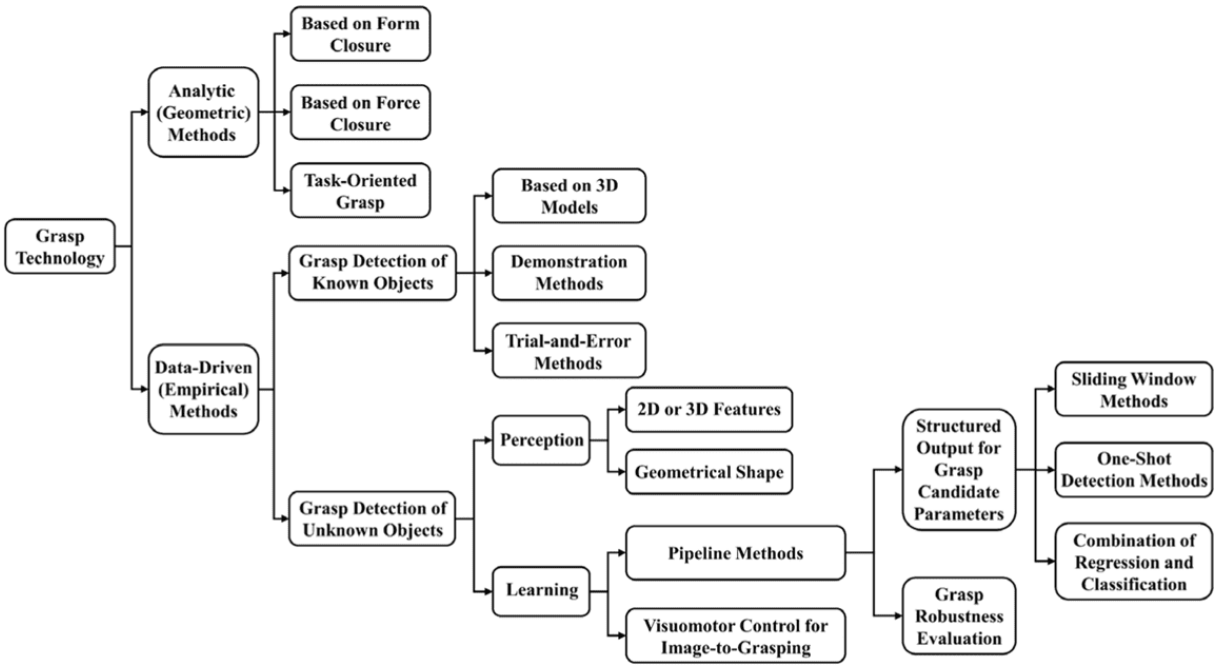


Figure 2.6: Exhaustive taxonomy of grasp detection methods taken from [12].

2.3 Grasp Detection

As discussed in detail in [11]–[13], [44], grasp detection techniques are responsible for predicting a set of possible grasps for an object in a given scene. The objective of these techniques is to facilitate the picking up of target objects and estimate its pose. To get a better overview of the grasp detection methods that will be explored in this section, the taxonomy depicted in Figure 2.6 can be used as a reference.

Grasp detection technologies can be classified into two main categories: analytic, also called geometric, and data-driven, also called empirical [12]. In the former category, grasps are found by analyzing the kinematic and dynamic modeling of the grasp operation and the physical characteristics of the object, such as its geometry and motion state. In contrast, the latter employs machine learning algorithms to generate gripper poses, thanks to the availability of large amount of data.

Moreover, data-driven methods can also be categorized as model-based or model-free, depending on the available knowledge regarding the object [12]. Model-free technologies are the ones suitable for an object-agnostic handover task and can be further classified into perception-based and learning-based approaches. The former focuses on identifying the geometric structure of data to generate and rank grasp candidates, whereas the latter aims to leverage machine learning-based methods to directly generate a set of robotic grasps.

In the case of unknown objects, learning-based approaches can be divided into two categories: pipeline methods, which distinguish between the grasp generation and path planning

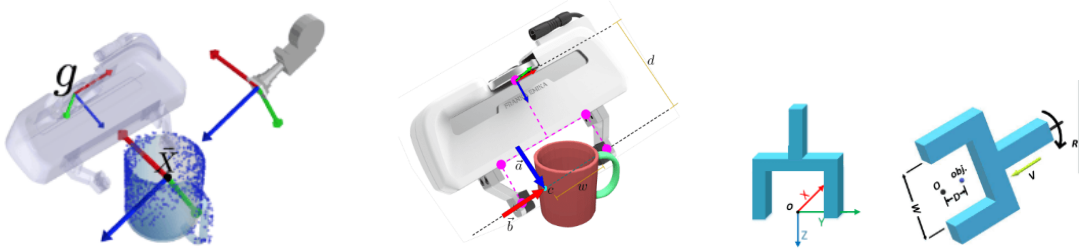
Figure 2.7: Both images show the 5-DoF rectangles used to represent a grasp in the image plane. On the left, the one considered by [30]. On the right the one by [47].

phases, and end-to-end methods, which directly map from image data to grasp action [12]. Pipeline methods, like [2], [3], [32], output either a robustness evaluation for the grasps, such as their probability of success, or a structured output, that directly identifies the grasp pose. In contrast to the previous approach, end-to-end methods learn visuomotor control policies directly from the input, in an image-to-action manner. For example, [45] directly assigns a score to each pixel of the input image and associates it to a grasping primitive action.

The input to these methods is typically RGB-D images, such as in [29], [30], [46], or partial point clouds, such as in [2]–[4], [23], [31]–[33], depending on how grasps are represented.

Depending on the implemented technique and the given input data, which is usually provided by an RGB-D sensor, grasp poses can be represented in different ways. Some techniques infer them directly from an RGB image, with the additional knowledge provided by the depth sensor, such as [29], [30], [46]–[48]. Other require the point cloud representation for the 3D data, such as [2]–[4], [23].

A common representation, which can be displayed as a rectangle on the image plane, as shown in Figure 2.7, is in the form of 7-DoF objects [30]. This representation consists of the pose the manipulator arm must assume to pick up the object. In this case, considering as reference the image, a grasp pose rectangle can be specified by 5-DoF object, consisting of (r_g, c_g, n_g, m_g, θ_g), where r_g and c_g refer to the upper-left corner of the rectangle, n_g and m_g are the dimension of the rectangle and θ_g the angle between the first edge and the image plane x-axis. The other 2 parameters are related to the distance of the center of the rectangle from the camera, z_g, and the opening width w of a parallel-jaw gripper. These 7 parameters allow for a full representation of the grasp pose. The translation vector t and the orientation θ with respect to the camera frame the robotic arm needs to assume to grasp the object can be retrieved from the computed parameters. Alternatively, they can be represented as a simplification of the previous representation as 5-DoF objects [48]. In this case, it is implicitly assumed that a 2D grasp can be projected back



(a) (b) (c)

Figure 2.8: From left to right, how grasps are represented in a 3D space in [3], [4], and [2]. Images taken from their corresponding paper.

to 3D and executed by the robot viewing the scene. Therefore, the only necessary information for representing a grasp would be analogous to an object detection problem with the addition of the grasp orientation. The needed parameters are (x, y, $width$, $height$, θ), where x and y are the coordinates of the rectangle's center, $width$ and $height$ its size, representing also the opening width of a gripper, and θ its orientation with respect to the camera frame [47]. Such representations constrain the gripper to approach the object from an orthogonal direction with respect to the image plane, limiting the diversity of potential grasps that can be detected. In fact, early works in grasp detection were inspired by object detection techniques, focusing on finding top-down grasps [13].

To overcome this issue, grasp poses can be directly predicted on the partial point cloud and represented in $SE(3)$, as in [32] or [3]. The first method, GPD, generates 6-DoF grasps in a sampling-evaluation manner, treating grasp detection as an object detection task, with a set of grasp candidates being sampled on the observed scene, according to some predefined conditions, and evaluated using an end-to-end approach. Similarly, PointNetGPD [33] employs PointNet [42] to evaluate a set of heuristically generated grasps. The second, 6-DoF GraspNet, whose grasp representation is shown in Figure 2.8a, uses a two-step pipeline comprising a generative approach for sampling a set of potential grasps, followed by an evaluation phase for rejecting implausible ones and, simultaneously, a refinement step for improving the detection. 6-DoF GraspNet employs the variational auto-encoder architecture [49], with both the encoder and decoder based on PointNet++ [43] and the grasp evaluation based on PointNet [42].

An efficient grasp detection model is GraspNet [2], which is trained on the GraspNet-1Billion dataset [2], comprising more than 1-billion grasp poses generated on a set of 88 objects. Grasp poses are represented as shown in Figure 2.8c and are defined with respect to the camera frame. Their analytical representation consists of the following matrix: $\mathbf{G} = [\mathbf{R}\ \mathbf{t}\ w]$, where $\mathbf{R} \in R^{3x3}$ denotes the rotation matrix associated to the gripper orientation, $\mathbf{t} \in R^{3x1}$ the translation vector associated with the grasp center position, and $w \in R$ the end-effector width necessary for picking up the target object. Since generating rotation matrices is a challenging

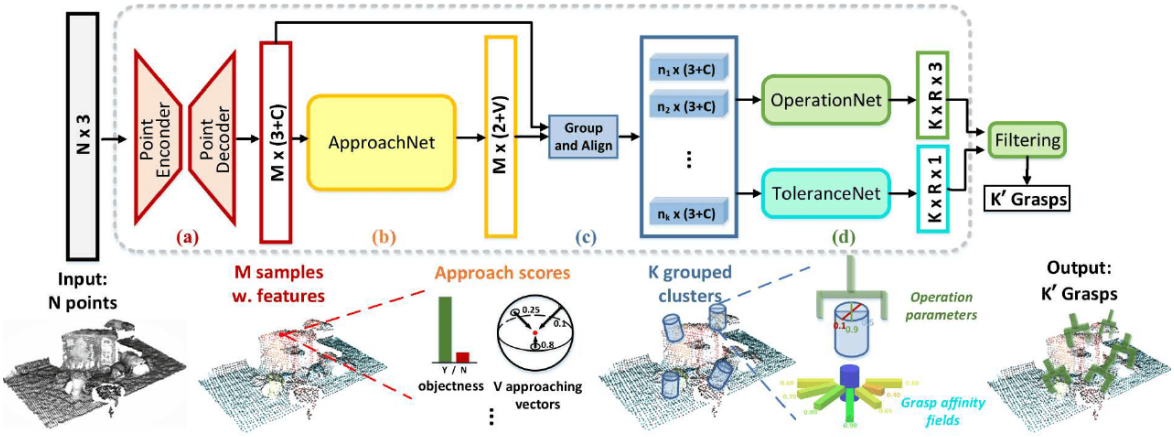


Figure 2.9: Overview of the GraspNet [2] model. The model comprises three sub-networks, ApproachNet, OperationNet, and ToleranceNet. ApproachNet is used to infer vectors of the grasps. OperationNet to infer the orientation that the gripper must have to pick up the object. ToleranceNet to make the grasps robust against noise. Image taken from [2].

task for neural networks due to their geometric properties, GraspNet first generates the position of the grasps and then infers the rotation associated with a viewpoint. The pipeline to generate grasps comprises three main phases, as shown in Figure 2.9: the Approach Network, which is based on PointNet++ [43], to infer approaching vectors and feasible grasp positions; the Operational Network, which is responsible for predicting in-plane rotation, approaching distance, gripper width, and grasp confidence; and the Tolerance Network, that predicts the tolerance to perturbation for each grasp pose. GraspNet analytically computes a score to each potential grasp according to the force-closure metric, as in [33]. In particular, given a grasp pose, the object point cloud and a friction coefficient, it assigns a binary label indicating whether the grasp is antipodal or not [50]. Then, the resulting score is computed according to

$$s = 1.1 - \mu,$$

where μ is the friction coefficient which is gradually decreased from 1 to 0.1. The grasp with lower friction coefficient is the one with higher probability of success.

Another model is Contact-GraspNet[4], which proposes a 4-DoF grasp and represent it as in Figure 2.8b. This method is based on the assumption that, given a point cloud, at least one of the two contact points of a parallel-yaw gripper is visible. Therefore, the only parameters to estimate are those related to the rotation and to the gripper width, with the additional advantage of facilitating the learning process. Each of the produced grasps is scored with a two-step process: (1) potential contact points for a successful grasps are found and scored, then (2) a network predicts the ideal placement and configuration of the end-effector for achieving a stable grasp.

The output of all the networks, however, can be easily represented in $SE(3)$, thus as a transformation matrix with respect to the camera frame. They used a different representation for the grasps to facilitate the learning process, thus reducing the number of parameters.

It is noteworthy that all the methods seen so far have one important thing in common: they generate grasps starting from static objects placed on a table. It is possible to utilize them even with this limitation, adapting them to the situation.

One promising state-of-the-art method for overcoming this issue is AnyGrasp [23], which incorporates center-of-mass knowledge of the object and the spatial-temporal domain for ensuring both temporal continuity and grasp quality. This method can generate dense and temporally consistent 7-DoF grasp poses for moving objects, using as input partial point clouds. The algorithm comprises two modules: a geometry processing module, which samples grasps given a partial point cloud, and a temporal association module, which enables grasp pose tracking across consecutive frames, allowing to deal with moving objects.

A different approach for the grasp generation on generic objects is the one proposed by [31], which consists of applying a point cloud shape simplification. In this case, point clouds are simplified into a collection of simpler shapes, such as cylinders, spheres, ellipsoid, or parallelepipeds, and grasps are sampled on this new representation.

Due to the prevalence of models which are trained on static objects placed on a flat surface, these networks need to be adapted to a more dynamic environment for tasks such as the H2R handover. In the context of this thesis, GraspNet [2], 6-DoF GraspNet [3], and Contact-GraspNet [4] are implemented and adapted for a human-robot handovering, as explained in Chapter 3.

2.4 Dataset and Evaluation Metrics

The objective of grasp detection technologies is to infer a diverse set of grasp poses from the RGB-D data representing the object. As previously described, grasps can be represented in two forms: as rectangles on the image plane or as poses in the 3D space.

In order to ensure that a model generalizes well on unseen data, the datasets should be sufficiently large and include a variety of different objects. In particular, grasps should be generated to cover the entire surface of a generic point cloud [2]. For the H2R handover task, the number of grasps should be large enough to allow the best one to be selected at a location that is not close to the body.

In the grasp detection techniques seen in the previous section, some methods construct their own dataset, such as [2], [3], [33], starting from a set of 3D models provided by ShapeNet [51], YCB [52] or DexNet [53]. While other, such as [4], [29], train their own network on already existing datasets, for example Graspnet-1Billion [2] or ACRONYM [54].

The generation of datasets typically requires the execution of multiple steps. One example is the generation process in [3], which, after sampling a set of grasps based on the object geometry, employs a physics simulation tool to validate them as successful or not. The simulation consists of testing the grasp robustness through the application of a shaking motion, keeping the surface friction and the object density properties constant. Another example is the generation of the GraspNet-1Billion [2] dataset, in which a set of grasps is sampled on the entire object surface and evaluated using a force-closure metric with different coefficients of friction. Only those grasps that are able to maintain the object with a low friction coefficient value are retained. After this step, a collision check is performed to prevent the generation of invalid poses. Moreover, a validation check is conducted to select only those grasps that allow a parallel-jaw end effector to pick the object.

With regard to the evaluation of human-robot interactions, there is a lack of standardised measurement tools and metrics for a fair comparison between different techniques [15], since their evaluation highly depends on the task to be performed. Moreover, the evaluation results may depend on the utilized number or type of objects, which can differ. To overcome this issue, some datasets, like YCB [52], suggest a collection of object that are readily available. Some commonly used metrics for measuring the overall performance are the success rate and the task completion time. While the former does not explain the causes of the issues, the latter depends on other factors, such as the chosen motion speed, particularly in the context of safety.

2.5 Human-To-Robot Handover Pipelines

In this section, some complete frameworks for the H2R handover task are analyzed.

The most common used vision system configurations are the *eye-to-hand* and *eye-in-hand*, with a single RGB-D camera. For what concern the type of gripper, a parallel-jaw one is usually considered, as shown in Figure 2.8. H2R handover tasks can be implemented following either an open-loop approach or a closed-loop one. In this context, the open-loop addresses the task of reaching the best grasp without dealing with potential hand motions. In contrast, the closed-loop one attempts to cope with them by updating the goal in case of motions.

The first revised approach, Rosenberger et al. [26], performed the handovering of 13 distinct objects of different sizes and shapes, with an *eye-in-hand* perception system. The presented approach comprises several steps, some of which are performed simultaneously. The initial step involves processing the RGB image and its depth in order to segment the body of the person, with particular attention paid to its hand, and to detect the held object. The segmentation masks and the bounding box of the object are then used to generate a set of grasps with the GG-CNN [46] model. Since the model has been trained on static objects positioned on a plane surface, a

virtual plane is placed behind the target object in order to simulate the same situation. As last step, the best generated grasp by the grasp detection model is executed. In this proposed pipeline, as highlighted in [26], the limitations are mainly attributed to the object detector module, which results in the object being not detected.

Another pipeline is the one proposed by Yang et al. [19], whose main objective was to investigate the preconditions and policies for a handover operation. In this case, the handover is performed with a single object, utilizing a restricted set of predefined grasps, which directly depend on how the object is held. The grasp detection model performs a simple pose classification, given as input an RGB-D image from an external camera. As described in [19], the limitations are mainly due to the presence of unseen grasp poses and to the lack of legible and friendly robot motions.

In their study, Liu et al. [28] presented a closed-loop approach for this task with an *eye-in-hand* RGB-D camera. The captured scene is used to perform a hand and object detection, followed by a segmentation of the hand. Then, the GraspNet [2] model is employed to infer a set of grasps on the object point cloud, retaining only those with a specific orientations with respect to the human body position. The pipeline was tested on 8 distinct objects. The observed limitations of this method [28] were the generation of unfeasible grasps, which were probably due to the lack of good-quality point cloud data, and the low number of points due to an excessive occlusion of the hand.

Finally, Yang et al. [27] presented a pipeline whose objective is to grasp generic objects. In particular, the implementation was tested on 26 different objects, with a perception system setup comprising an external RGB-D camera. Given the produced inputs, a hand segmentation module is used to obtain the corresponding mask from the RGB image, while the depth is used to extract the point cloud of the hand holding an object. Then, grasps are detected only on the object point cloud using the GraspNet [2] model. Since GraspNet is trained on static objects, it has been adapted to a dynamic context. To ensure temporal consistency over consecutive frames, a perturbation is applied to the generated grasps at each time stamp, retaining only those grasps that lead to an improvement in the score. Otherwise, if the score is lower, perturbed grasps are accepted with a probability that depends on the new and old scores. They demonstrated that, in this way, grasps are more stable and consistent over time, maintaining a relatively constant pose with respect to the objects. Moreover, in their experiments, they assumed that the human will adapt to the robot's motion. The observed limitations, as described in [27], were due to: the high computational costs required to continuously update the target grasp to maintain a closed-loop approach; the lack of some depth information due to the object properties, i.e. dark surface; some segmentation problems, where the hand was considered to be part of the object; the noise of the point cloud due to nearby objects.

Human-2-Robot handover is a task that involves various fields of study, especially coming from computer vision and robotics. For this reason, the necessity for the development of reliable and efficient segmentation and grasp generation technologies plays an important role. The implementation of a complete pipeline for the H2R handover evidences a series of different problems from both the aforementioned fields of study. These include the difficulty of detecting feasible grasps due to a noisy point cloud, the difficulty of segmenting the object correctly due to its physical characteristics, the high computation cost for a closed-loop approach, and the lack of legible and friendly robot motions for a reliable handover. The handover task has been implemented by Rosenberger et al. [26], Yang et al. [19] and Liu et al. [28] following an open-loop approach, while Yang et al. [27] has considered a closed-loop approach.

Furthermore, the absence of a uniform evaluation metric for this kind of task makes it challenging to compare different implementations, either in terms of robustness of grasping generic objects and safety.

Chapter 3

Proposed Pipeline for H2R handover

In this chapter, a proposed solution to the H2R handover problem is presented, which can be seen as a three-step procedure, as in Figure 3.1. The initial step exploits a segmentation module to distinguish between the hand and the object from other elements captured by the camera. Two approaches, namely, *Fast-EgoHOS* and *Complete-EgoHOS* are described. This is followed by a grasp generation module to detect a set of gripper poses on the given object point cloud. In this phase, three different models described, namely, GraspNet [2], Contact-GraspNet [4], and 6-DoF GraspNet [3]. Finally, a selection module is employed to choose and move the robot arm towards the optimal grasp. In this phase, three discarding criteria are defined.

The considered robot configuration for the implementation consists of an RGB-D camera mounted on the robot's end effector.

3.1 Segmentation Module

The segmentation module is responsible for distinguishing between the human hand and the object being held. In the context of the H2R handover task, the segmentation mask is useful for enabling the system to localize the specific region of the image that contains the hand and which is associated with the object. This allows for the removal of grasps that may result in the robot colliding or pinching the hand.

One of the goal of this thesis is to achieve a good trade-off between accuracy and processing speed for the entire handover pipeline. For this purpose, since the grasp generation module is the most onerous one in terms of execution time, reducing the processing time where possible is necessary. To achieve this result, the EgoHOS [1] model is selected, mainly for two reasons. Firstly, it is specifically suited for our scenario where the camera has an egocentric view with respect to the human body, and thus to the hand and the object. In this case, the egocentric view is from the point of view of the robot, which is similar to the one adopted by [1] to train the

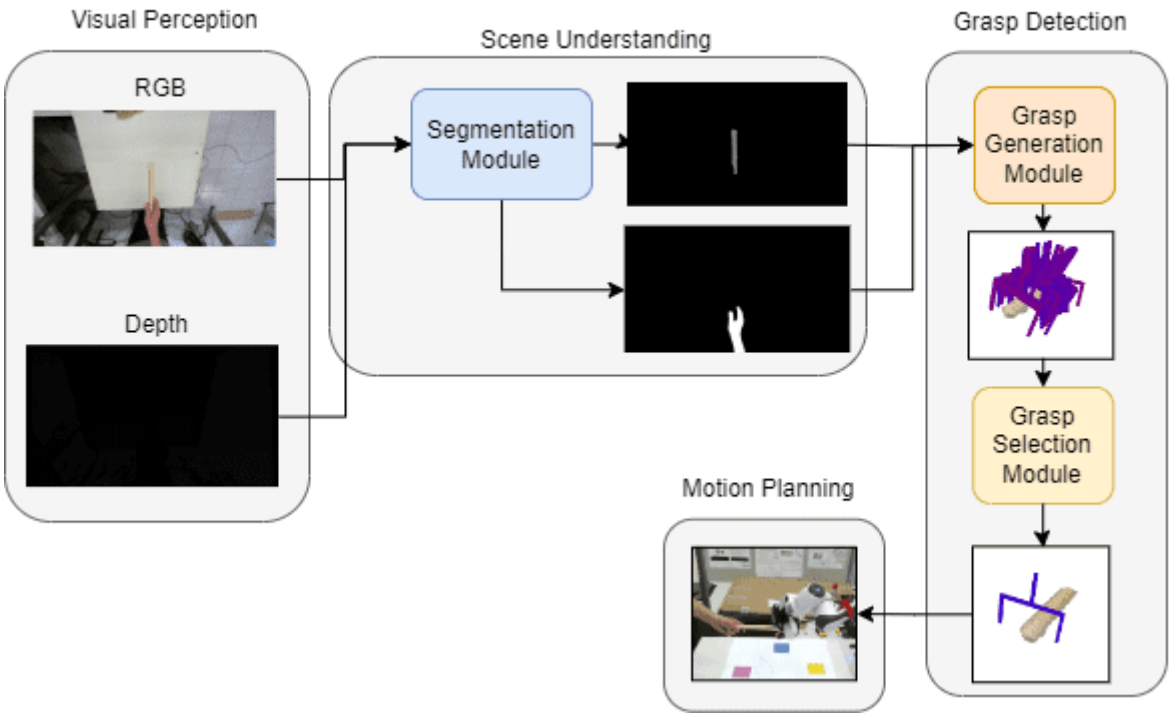


Figure 3.1: This scheme shows the main parts of the proposed pipeline for the H2R handover task.

segmentation model. Secondly, this model is designed for a per-pixel segmentation of hands and generic objects with which the person is interacting. Segmenting a generic object is crucial in our scenario, since the main purpose is to enable a robotic arm to pick up any object from a human's hand.

As shown in Section 2.1, the EgoHOS pipeline consists of three subsequent steps, each exploiting a different model and receiving as input the initial RGB image concatenated with the previous step outputs. The first step consists of segmenting the left and/or the right hands that could be present in the image, without taking into consideration any object. Then, since directly segmenting a generic object can capture background clutter, a second step is needed to focus on inferring the contact region, called *dense contact boundary* [1], between the hands and the objects. This phase helps improve the segmentation accuracy providing a cue to discriminate between the many background objects in the input image and the important objects. Finally, the last step consists of predicting the object mask with the knowledge given by the previous steps. A result of this pipeline is shown in Figure 3.2.

As mentioned at the beginning of this section, one of the key aspects to keep into consideration is the trade-off between accuracy and speed. To this end, two solutions are proposed: one that is more accurate but slower, which will be called *Complete-EgoHOS*, and the other that is

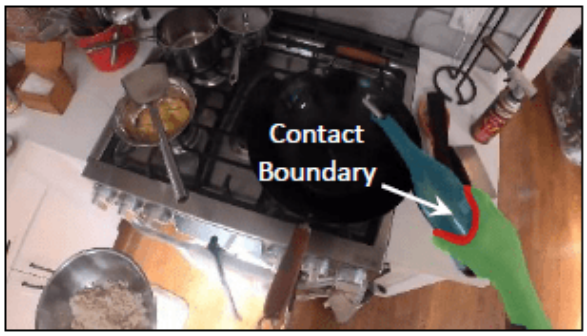


Figure 3.2: The image shows the EgoHOS result for an input image. It consists of a segmentation mask of the hand, the contact boundary and the interacting object.

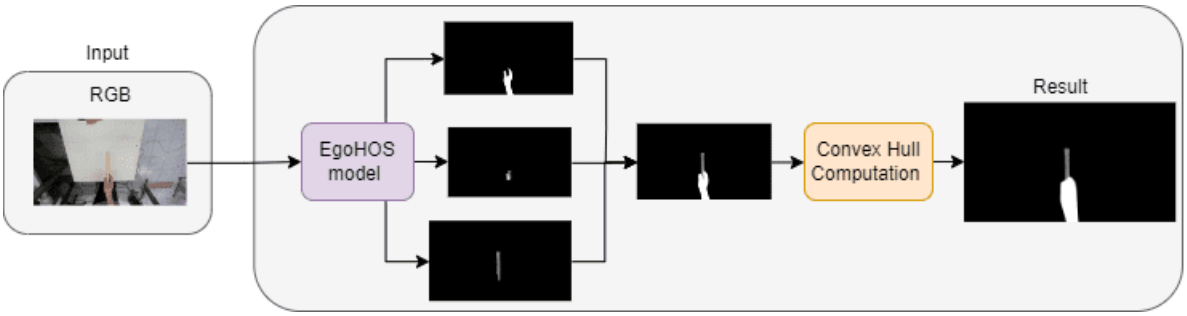


Figure 3.3: This scheme shows how the segmentation is done for the *Complete-EgoHOS* methodology.

less accurate but faster, called *Fast-EgoHOS*.

The first proposed solution is based on directly inferring the hand and the object with the complete implementation of the EgoHOS model, as shown in Figure 3.3. Since the model is trained for segmenting hands along with their interacting objects and contact boundary, two main advantages have been observed.

Instead, the main idea behind the second one consists of using only the initial part of the EgoHOS pipeline, specifically the hand segmentation part, while relegating the object segmentation to a post-processing stage. The reason for this choice is to accelerate the segmentation process, with a compromise in terms of accuracy in exchange for a real-time approach.

The input data is provided by the only sensor available in our setup. The RGB image feeds the segmentation model to infer the hand mask, while the depth is used as input for the post-processing part. The proposed method leverages the assumption of a single hand-object interaction. While this assumption simplifies the problem, it may not be suitable for all scenarios where there are multiple hands and clutter.

As illustrated in Figure 3.4, the proposed post-processing method is as follows. Given the hand mask, the mean distance between the camera and the hand is computed, allowing the estimation of how far the hand and the object are. To retrieve the object mask, only the depth

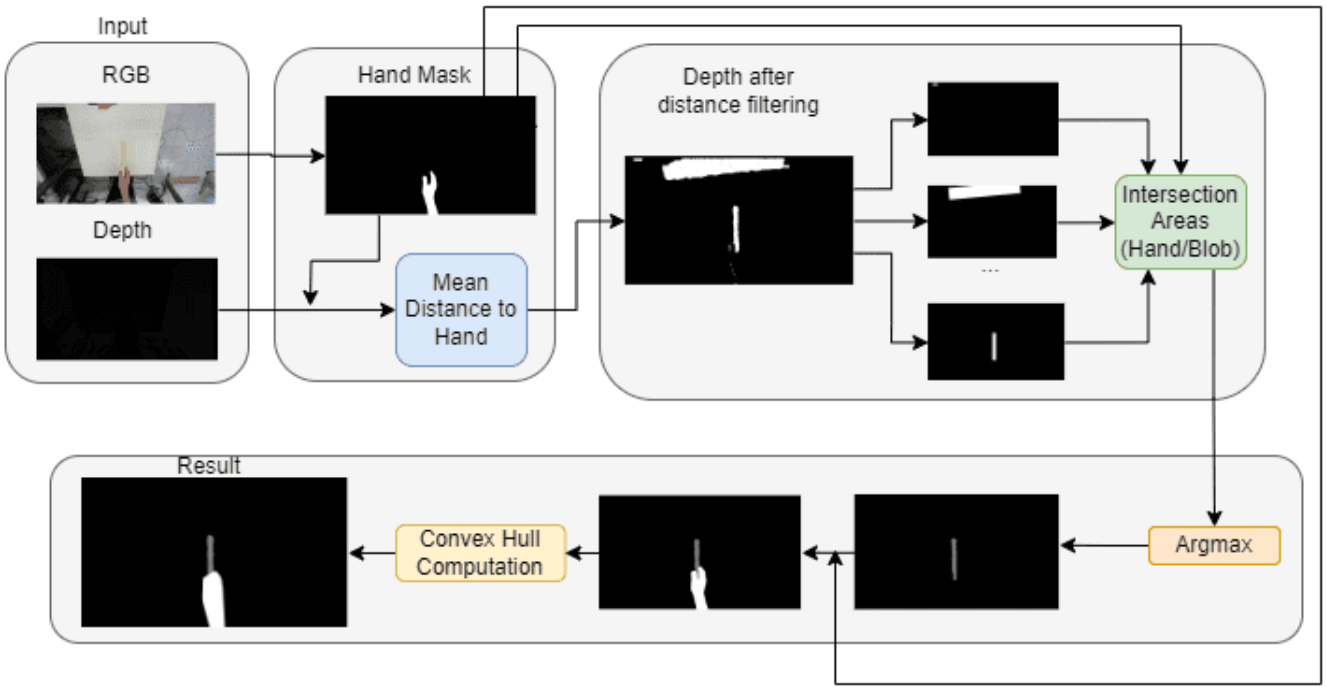


Figure 3.4: This scheme shows how the segmentation is done for the *Fast-EgoHOS* methodology.

values which are not associated with the hand and fall within two distance-based thresholds are kept. This range, whose values directly depend on the previously computed mean distance, can be visualized as two virtual planes, one behind the object and one in front of it. In this way, the only values that are kept are the ones related to any object whose depth is within the specified thresholds.

Then, since there may be outliers, it is necessary to remove the remaining depth values that are not associated to the correct object. For this to be achieved, the intersection area between a bounding box around the hand's convex hull and a bounding box around each potential object's blob is computed. Then, leveraging the aforementioned assumption, the blob with the largest intersection area is selected, which is likely the one associated with the manipulated object. After this procedure, the object mask is associated to the blob with the largest intersection area, discarding all the other.

The common post-processing operation between the two solutions is the dilation and representation as a convex hull of the hand mask. The convex hull is necessary for merging different parts of the hand in case of occlusions, while the dilation is applied to avoid any values near the hand from being misclassified as object. Finally, starting from the object mask, an estimate of the length of the object is computed. This value will be used on Section 3.3 to discard some of the grasps.

The main distinction in terms of outcomes between the two approaches follows the trade-off

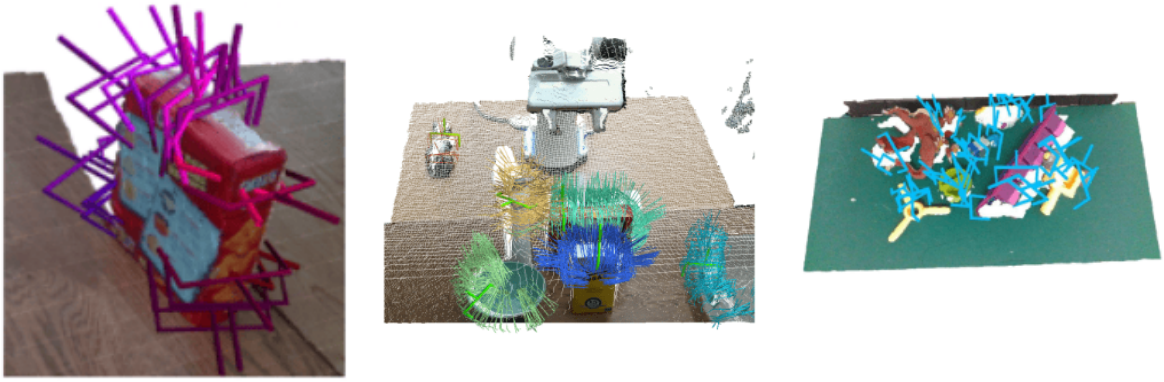


Figure 3.5: These images show the point cloud that is expected from the models. In particular they expect to receive a plane surface with objects laying on it. From left to right, images are taken from [3], [4] and [2].

between accuracy and processing speed, as demonstrated in Section 4.2.1.

Moreover, in the first segmentation method there is no need for the post-processing part to understand which blob is the object, since only an RGB image is required as input.

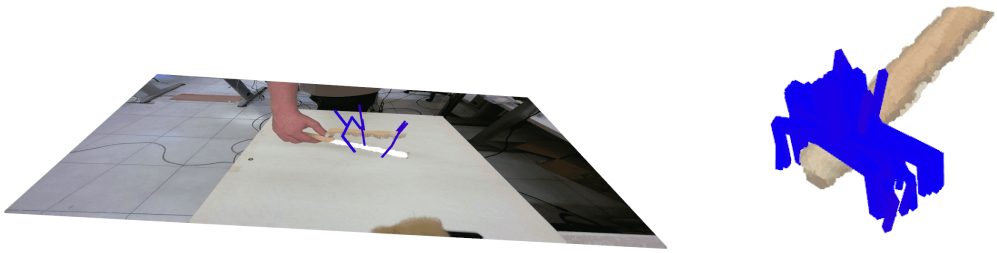
3.2 Grasp Generation Module

The grasp generation module has the objective to generate a set of potential grasps for the object. It is important to highlight that this thesis focuses only on the data-driven case, in which the object model is unknown and a learning method is exploited. Therefore, the following three models to generate grasps starting from a point cloud have been considered: GraspNet [2], 6DoF-GraspNet [3], and Contact-GraspNet [4]. Moreover, the choice of these grasp planners was driven by the fact that they are model-free and efficient.

The hand-object mask, produced in the previous step, is received by this module, where a common pre-processing procedure is applied to the data. In particular, the knowledge of the hand-object mask is exploited to remove from the depth map the information related to the hand and the background, thus distinguishing the elements not directly related to the object.

Input and output of the three chosen models are the same: a point cloud of the scene or of the object only and a set of grasps represented in $SE(3)$, respectively. In particular, each grasp is represented as a transformation matrix $\mathbf{g} = [\mathbf{R}\ \mathbf{t}]$ with respect to the camera frame, where $\mathbf{R} \in R^{3x3}$ denotes the rotation matrix associated to the gripper orientation and $\mathbf{t} \in R^{3x1}$ the translation vector associated with its center position. For further details, see Section 2.3.

It is important to notice that these models were developed for generating a set of grasps on a collection of objects positioned on a plane surface, i.e. a table. In this case, however, there is a significant distinction: objects are not laying on a surface. As illustrated in Figure 3.5, models



(a) (b)

Figure 3.6: Examples of the two approaches used to process the input point clouds. On the left image the approach involving the imitation of a plane; on the right image the approach which directly isolates the object point cloud.

have been trained on point clouds consisting of a set of objects on a table. Consequently, it was necessary to adapt the provided input to the requirements of the networks.

Two distinct approaches have been investigated and evaluated for adapting it, both exploiting the knowledge given by the hand-object mask. The first method, shown in Figure 3.6a, consists of creating a virtual plane situated below the object, flattening all the depth data not related to it, thereby imitating a flat surface. Instead, the second, shown in Figure 3.6b, directly removes all the surrounding elements, isolating the object point cloud.

The creation of a virtual plane has been demonstrated, in Chapter 4, to be an effective approach when applied to the input data of GraspNet. While, the isolation of the object point cloud was evaluated for either the Contact-GraspNet, GraspNet and the 6-DoF GraspNet models. Both adaptation approaches allow the models to produce a dense set of suitable grasp poses positioned on the entire object's surface.

3.3 Grasp Selection Module

The grasp selection module is responsible for selecting which of the generated grasps are suitable for execution by the robot. In particular, a filtering mechanism must be applied to remove the grasps that may be infeasible for the robot to reach or may result in the robot colliding with the person. Therefore, the filter is applied with the objective to discard those grasp candidates that could potentially hit the person or wrongly pick the object.

Given the results of the previous modules, namely the set of grasps, the hand-object mask, the mean distance, and the estimated object length, it is necessary to select which of the received grasps are suitable for the handover. For this reason, the applied safety policy is based on the implementation of three criteria:

1. Hand-grasp distance, in which grasps are discarded according to their distance from the

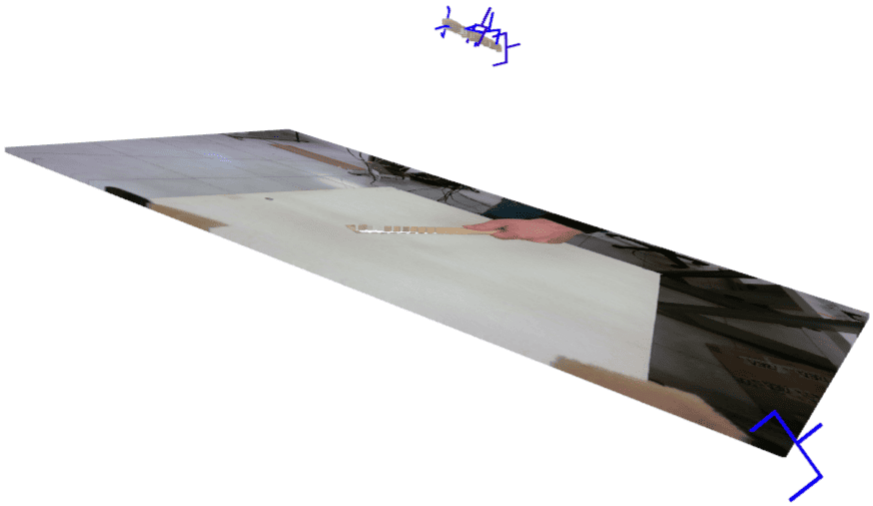


Figure 3.7: Problem that could arise if grasps not satisfying the first criterion were allowed.

hand. The hand-object segmentation mask is used to achieve this;

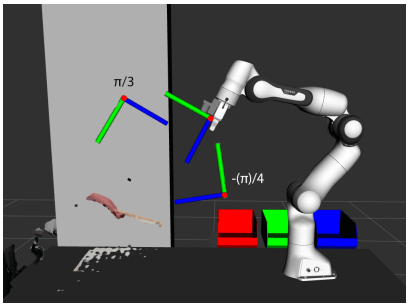
2. Object-grasp distance, in which grasps that are not close to the object are discarded, as the one in Figure 3.7;

3. Grasp orientation, in which grasps with an orientation that may cause the end effector to collide with the object or the person are removed, as shown in Figure 3.8.

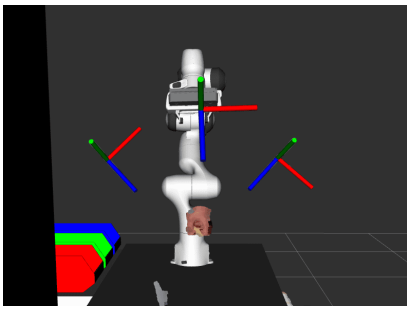
In the first criterion, grasps are discarded according to their distance from the hand. In particular, only those grasps whose distance is larger than the average distance between each grasp and the projected hand mask are selected. The projection was computed based on the mean distance, calculated in Section 3.1, rather than the corresponding depth value according to the mask. In fact, values not associated with the hand could be included if the dilated hand mask was considered, resulting in an inaccurate projection.

For the second criterion, all the outlier grasps that are not close to the object are discarded. For this purpose, they are removed if their position is not within a threshold which is based on the mean distance computed during the segmentation process. This criterion is useful in case a virtual plane is placed behind the object, in fact, some grasps may be generated on the plane border, as can be observed in Figure 3.7. The main issue arises when that grasp is the only one available or is the one with higher score.

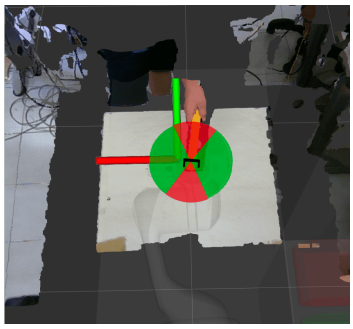
Finally, in the third criterion, all grasps whose orientation, with respect to the camera frame, causes the end effector to collide with the object or the person are deleted. To accomplish this,



(a) Roll range.



(b) Pitch range.



(c) Yaw range.

Figure 3.8: These three images show the allowed ranges for the roll, pitch, and yaw angles. Figure 3.8c, is displayed with a top view of the robot. The green areas are allowed ones.

three ranges are computed for the roll, pitch, and yaw angles, respectively. Since the robot is assumed to have a constant initial configuration, the roll and pitch ranges are fixed, while the yaw range depends on the object orientation. In particular, the yaw range is calculated according to the orientation of the longest side of the object with respect to the camera, allowing only grasps nearly perpendicular to the object to be selected. The configuration of these ranges allows to discard grasps whose orientations result in the robot assuming poses that are in proximity to the person or that interpenetrate the object, as shown in Figure 3.9.

In the current implementation, the grasp detection and selection modules are executed mul-

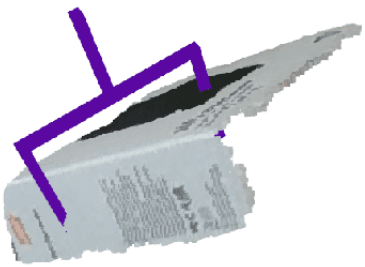


Figure 3.9: Problem that could arise due to the point cloud quality. This type of grasp is not satisfying the third criterion.

tiple times to collect a sequence of potential grasp candidates on successive frames. This is done to obtain a greater number of grasps from which to select if the proposed criteria result in the elimination of a significant number of candidates.

To prevent grasps from being located far from where the handover should occur, i.e. the final position of the hand, grasps from consecutive frames are compared and discarded according to their distance in the space. For example, let us consider the case where a set of grasps is detected when the hand and the object have just entered the camera's field of view, and another set is detected when the hand is in its final position. If the distance between the best grasps is larger than a certain threshold, defined based on the estimated object length computed in Section 3.1, then all the grasps should be recomputed for the successive new frames.

The last step of this module consists of selecting which of the remaining grasps of the sequence should be executed. Since the grasp generation models provide a score, the grasp with the higher one is chosen for execution.

Chapter 4

Experimental Evaluation

The objective of this chapter is to perform an experimental evaluation of the entire implemented procedure. To test the modules, two different evaluations are performed: an offline evaluation and an online one. A first evaluation is conducted with the aim of selecting which pair of models for the segmentation and grasp detection perform better according to the time-accuracy trade-off. The segmentation models are evaluated in terms of Intersection-Over-Union (IoU) and Average Execution Time (AET) on a previously recorded set of data comprising images of an hand holding an object. While, the grasp detection models are evaluated according to the number of generated grasps and how many of them are discarded, based on the criteria described in Chapter 3. The selected segmentation-grasp detection pair should exhibit good performance both in terms of time, performance and grasp generation.

A second evaluation is conducted on the physical robot to validate the efficacy of the previously selected pair of models a set of generic objects.

The evaluation is organized as follows. Section 4.1 describes the experimental setup, the hardware involved and the protocol used in the experiments. Section 4.2 analyzes the performance of the implemented methodology through an offline evaluation, while Section 4.3 presents the results obtained by testing with the robot. Finally, Section 4.4 analyzes the encountered difficulties and limitations of the current implementation.

4.1 System and Experiments Setups

The experiments were conducted with the 7-DoF Franka-Emika Panda robot, shown in Figure 4.1, mounted on a table with a two-finger parallel-jaw gripper as end-effector and MoveIt! [55] as motion planner. To increase the friction, thus the grasp robustness, two pieces of foam rubber were attached to it. The perception system is constituted by an Intel RealSense D455 RGB-D camera mounted on the robot's end-effector, which allows for an egocentric point of view of

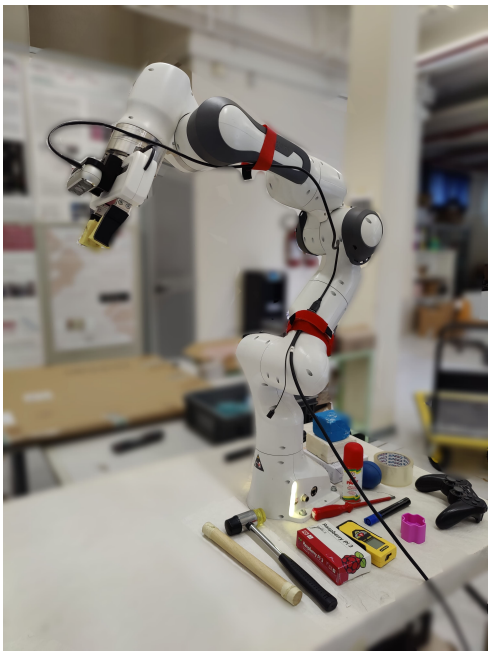


Figure 4.1: 7-DoF Franka-Emika Panda robot used for the experiments.

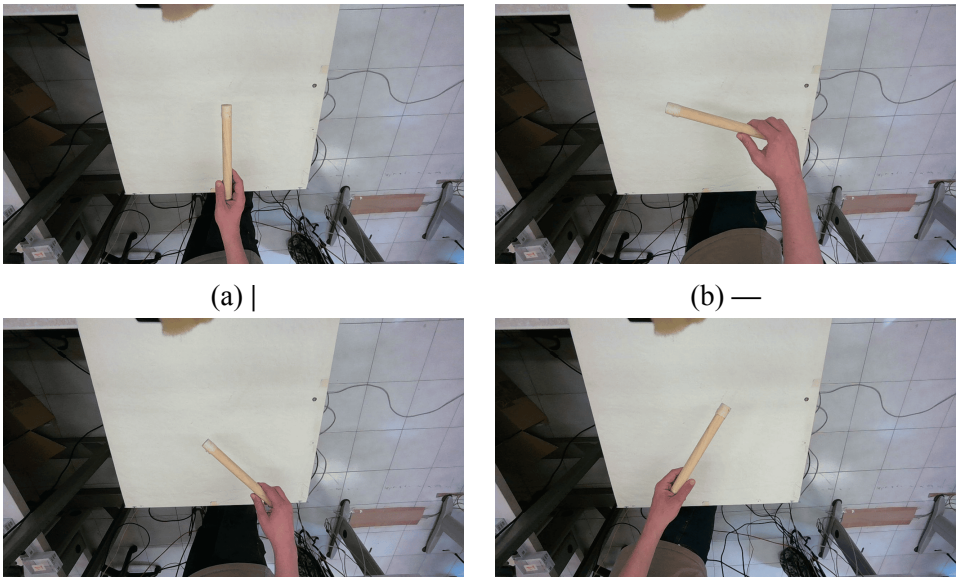
the scene. This is captured with a resolution of 1280x720 at 30 FPS. For the experiments, the robot starts from a predefined pose in order to ensure that the object and the hand are within the camera's field of view. The segmentation and grasp generation models run simultaneously on an NVIDIA GeForce RTX 2080 GPU.

The objective of the experiment is the successful transfer of an object from the human hand to the robot. The handover operation consists of the following phases:

1. the segmentation module is employed to distinguish between hand and object in the image;
2. the grasp generation module is implemented to generate a set of valid grasp poses, given as input the segmentation mask;
3. the selection of a subset of grasps according to the criteria described in Section 3.3;
4. the control of the robot manipulator is employed to plan a trajectory to an approaching pose, which is defined such that the desired grasp pose can be reached by performing a straight motion;
5. the grasping operation is performed by closing the end-effector fingers, after the grasp pose has been reached;
6. the robot arm moves backward while grabbing the object.

To test the system, a collection of 15 different objects, shown in Figure 4.2, is selected. The objects have been selected to have different characteristics in terms of size, shape and materials

Figure 4.2: List of 15 objects used for testing. The set comprises a variety of different objects which mostly differ on their size and shape.



(a) |

(b) —

(c) \

(d) /

Figure 4.3: Example of the type of images in the dataset.

in order to validate both the segmentation and grasp detection modules. Some of the objects have a similar geometric shape, but different sizes. The objects were selected considering the maximum opening width of the gripper. For instance, the remote, the white rectangle, the Raspberry box, and the hard disk box share the main shape, which is rectangular, but with different thickness. The same shape is also shared between the spray can, the marker, the wood stick, and the red screwdriver, which is cylindrical. Objects such as the hammer and the spray have a reflective surface, which may cause segmentation problems. While, objects like the tape and the controller have a shape which may be challenging for grasp detection.

To validate its robustness in response to different types of handover, each object is presented to the robot from four different orientations, which are represented, in the results, with the following symbols: |, —, \, and /, as can be seen in Figure 4.3. The objects are held at one end, so that the robot can grasp them from the other end. Moreover, four of them have been handled in two different ways, according to their shape or size. For instance, rectangular objects, such as boxes, can be presented to the camera in either a horizontal or vertical orientation, and the tape can be handled in a manner that either shows the hole or does not. This difference, in the experimental results, is highlighted with an underscore in the object name, followed by *h* or *v*, which correspond to horizontal or vertical, respectively. In total, the are 76 distinct object configurations.

The implementation of the three modules, described in Chapter 3, requires some parameters to be chosen in advance, mostly on the basis of empirical evidence. The parameters that need to be chosen in the Segmentation Module, see Section 3.1, are the following: (1) the depth threshold distance beyond which depth data are ignored, that in our case has been set to 80 cm, and (2) the distance at which to place the two virtual planes with respect to the object, which has been set to 7 cm.

For what concern the Grasp Generation Module, see Section 3.2, the parameter to be chosen is the distance at which to place the virtual surface to simulate a table. This value can be fixed, that is selected in advance, or variable, thus depending on the mean distance of the hand, as in Figures 3.7 and 3.6a, respectively. In the former case, the table can be positioned far from the object at a predefined distance, whereas in the latter, the distance between the surface and the object can be adjusted either closer or farther, depending on the applied displacement.

Finally, the Grasp Selection Module, Section 3.3, requires the following parameters to be selected: (1) the window size, that is the number of set of grasps to consider before choosing the final one; (2) the distance, relative to the camera frame, above which erroneously generated grasp poses are discarded, i.e. if they are on the table or on outlier points of the point cloud, as can be seen in Section 4.2.1; (3) the threshold distance under which to discard grasps, that has been set to be the average between each grasp and the hand; (4) the allowed range of values

of roll, pitch and yaw angles with respect to the camera frame, which have been empirically set. As can be observed in Chapter 3 at Figure 3.8, roll and pitch have been chosen to ensure that the robot does not assume poses under the object or with the arm on the same side of the person, whereas the yaw value has been chosen such that to force the end-effector to assume poses perpendicularly to the object orientation. The ranges have been defined as follows: the roll interval has been set to $[-\pi/4, \pi/3]$; the pitch range to $[-\pi/4, \pi/4]$; the yaw ranges to $[\theta + \pi/6, \theta + 5\pi/6]$ and $[\theta - 5\pi/6, \theta - \pi/6]$, given θ the orientation of the the object with respect to the camera frame. The yaw interval covers a $120°$ area on either sides of the object.

4.2 Offline Evaluation

The offline evaluation consists of testing the performance of both the segmentation and grasp detection modules with a collection of recorded data. This set comprises a sequence of approximately 50 to 60 images for each pose of the objects, for a total of 4013 frames. As shown in Figure 4.3, the images depict a hand holding an object directly in front of the camera.

4.2.1 Segmentation Module Evaluation

In order to evaluate the two proposed solutions for the hand-object segmentation, namely *Fast-EgoHOS* and *Complete-EgoHOS*, two metrics have been selected: the mean Intersection-Over-Union (IoU) and the Average Execution Time (AET). The IoU consists of measuring the overlap between a ground truth mask and a predicted one. This metric has been selected to quantitatively measure the object segmentation, comparing the ground truth masks, found using the *Segment Anything tool* [56], with the inferred object masks. Since the segmentation focuses on the object mask, the ground truths have been obtained by segmenting only the object being held. Instead, the AET measures the time required by each solution to infer the hand-object mask.

The objective of computing these two metrics is to have a comparison of the two methods, selecting which offers a good balance between accuracy and execution time.

In order to compute the IoU, a subset of the images was chosen, with one frame chosen every five, resulting in a total of 760 images, with 10 images per pose. While, the AET was computed averaging over the inference time for each image of the dataset.

From Tables 4.1 and 4.2, it is possible to observe the obtained results for the two metrics. The second method, which does not include the post processing phase, has shown better performance in terms of IoU, while the first one, has shown better AET. It is important to mention that, in case of the first method, the inference time is the average time needed for only inferring the hand mask, while for the second, it also includes the inferencing of the *contact boundary* and

Object	IoU *Fast-EgoHOS*	IoU *Complete-EgoHOS*	Mean Difference
hard_disk_box_h	0.8961	**0.9142**	0.0181
hard_disk_box_v	0.7728	**0.8987**	0.1259
ball	0.8233	**0.8355**	0.122
controller	0.8858	**0.9123**	0.0265
blue_cube	0.83	**0.8868**	0.0568
hammer	0.7172	**0.8189**	0.1017
hand_cream	0.8531	**0.8686**	0.0155
marker	0.7	**0.7735**	0.0735
pink_object	0.773	**0.8237**	0.0507
raspberry_box_h	**0.8977**	0.8907	-0.007
raspberry_box_v	0.882	**0.9075**	0.0255
remote	0.7838	**0.8379**	0.0541
screwdriver	0.6829	**0.7021**	0.0192
spray	0.8490	**0.8775**	0.0285
tape_h	0.3301	**0.4968**	0.1667
tape_v	0.7995	**0.8608**	0.0613
white_rectangle_h	**0.8531**	0.7556	0.0975
white_rectangle_v	0.8554	**0.8592**	- 0.0038
wood	0.7901	**0.8210**	0.0309
Total:	0.788	**0.8284**	0.0404

Table 4.1: Comparison in terms of IoU between *Fast-EgoHOS* and *Complete-EgoHOS*. Values are expressed as mean IoU and are computed over a subset of the dataset, consisting of 40 images per object, 10 per pose. The last column shows the difference between *Complete-EgoHOS* results and the *Fast-EgoHOS* ones.

the object mask. Moreover, the processing time is the mean time needed to process the masks, as explained in Section 3.1, while the total time is the time needed to infer, process and publish the result. Moreover, the last column of 4.1 shows the difference in the mean values. It is possible to notice that *Fast-EgoHOS* reaches performance comparable with *Complete-EgoHOS*.

As already described, in the context of an handover, accuracy and execution time are two crucial aspects to consider. According to the obtained results, which compare two proposed solutions, both aspects have been considered. *Complete-EgoHOS* reached a mean IoU value of 0.8284, which is better than the one obtained by *Fast-EgoHOS*, that is 0.788. In the case of AET, the situation is inverted, with *Fast-EgoHOS* performing with an average time of 0.076 seconds per image, while *Complete-EgoHOS* took an order of magnitude more, with 0.22 seconds.

Based on the obtained results, the selected method for hand-object segmentation in the online evaluation is *Fast-EgoHOS*, as it offers a good balance between IoU and AET results.

Figure 4.4 shows the difference between the object masks provided by the two methods. In particular, it is evident that the segmentation inferred by *Fast-EgoHOS* is more fragmented and

	Inference [s]	Processing [s]	Total [s]
Fast-EgoHOS	**0.056**	0.02	**0.076**
Complete-EgoHOS	0.219	**0.01**	0.22

Table 4.2: Comparison in terms of AET between *Fast-EgoHOS* and *Complete-EgoHOS*.

(a) (b)

Figure 4.4: On the left the object mask found by *Fast-EgoHOS*; on the right the object mask inferred by *Complete-EgoHOS*.

less accurate with respect to the one provided by *Complete-EgoHOS*, as confirmed by Table 4.1. This difference is mainly due to the manner in which masks are extracted. In the first case, the mask is inferred starting from the knowledge of the hand mask and the depth, while in the second case, the mask is inferred by the neural network from the RGB image only. In practice, this difference is not a significant issue, especially for the grasp detection module. In this case, rather that utilizing the RGB image as input, grasp poses are generated from point clouds, which may be inaccurate; it is sufficient to have the contour of the correct object. It is important to notice that, as shown in Figure 4.5, some outlier points can be considered for the grasp detection in case of over-segmentation, specifically when the depth associated with a specific pixel is not excluded by the constraints. This issue has been observed only with the *Complete-EgoHOS* method, since it is image-based only.

4.2.2 Grasp Detection and Selection Evaluation

To evaluate these two modules, which are closely related, a set of grasps is firstly generated by the Grasp Detection Module and then processed by the Grasp Selector[1], according to the procedure described in Sections 3.2 and 3.3.

The evaluation of these modules is carried out in accordance with the following metrics: the number of frames that allows to generate at least one grasp, the quantity of generated grasps, the quantity of discarded grasps, according to the discarding criteria described in Section 3.3, and

[1] From now on, we refer to the set of grasps processed according to the criteria described in Section 3.3 as *pruned grasps*.

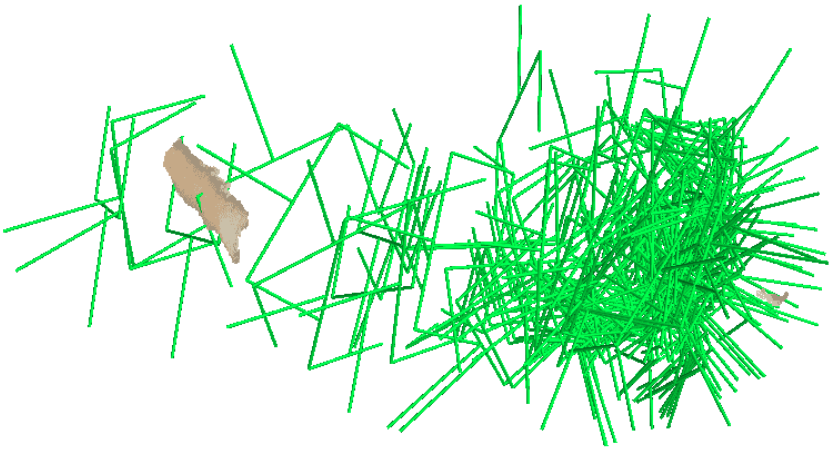


Figure 4.5: Problem that arises when an over-segmented mask is used to generate the object point cloud. From the image, it is possible to observe that grasps are generated on the object, located on the left, but also on some outlier points, on the right. The image shows only a subset of the total of the generated grasps.

the time required to convert the RGB-D data into a point cloud for the network, along with its inference time.

In addition, in order to analyze the distribution of generated and pruned grasps on the objects, the following values are considered: the mean distance between the best grasp, in terms of score, to the hand, D_{b_h}, and the mean distance between the best grasp and all the other grasps of the same set, D_{b_g}.

These two distance-based metrics are utilized to evaluate the ability of the grasp generation modules to generate a diverse set of grasps, specifically in this context. In particular, the aim is to verify whether the set of pruned grasps still contains grasps that are not in proximity to the hand or with a wrong orientation. To this end, D_{b_h} is used to compare the distance to the hand between the best grasp of the pruned set and the original set, whereas D_{b_g} is employed to verify how grasps are distributed with respect to the best ones.

In order to evaluate each metric, the hand-object masks generated by both *Fast-EgoHOS* and *Complete-EgoHOS* are utilized as input, along with the corresponding depth. Instead, grasps are evaluated using the following grasp detection models: GraspNet [2], Contact-GraspNet [4], and

6-DoF Graspnet [3]. They accept the object point cloud as input, however only GraspNet accepts a point cloud with a virtual surface behind the object. In this case, the distance at which to place the plane has been set at 1 mm along the camera z-axis. Although this value lacks a physical meaning, it enables the model to generate more consistent grasps than those generated with the object point cloud only, as can be observed in Figure 4.6. The figure illustrates the distinction between some grasp poses generated by GraspNet with and without the virtual plane. This behavior is probably due to the overall impact these points have on the computation.

The decision regarding the distance at which to place the virtual plane was based on empirical evidence, in line with the number and quality of grasps observed during preliminary experiments.

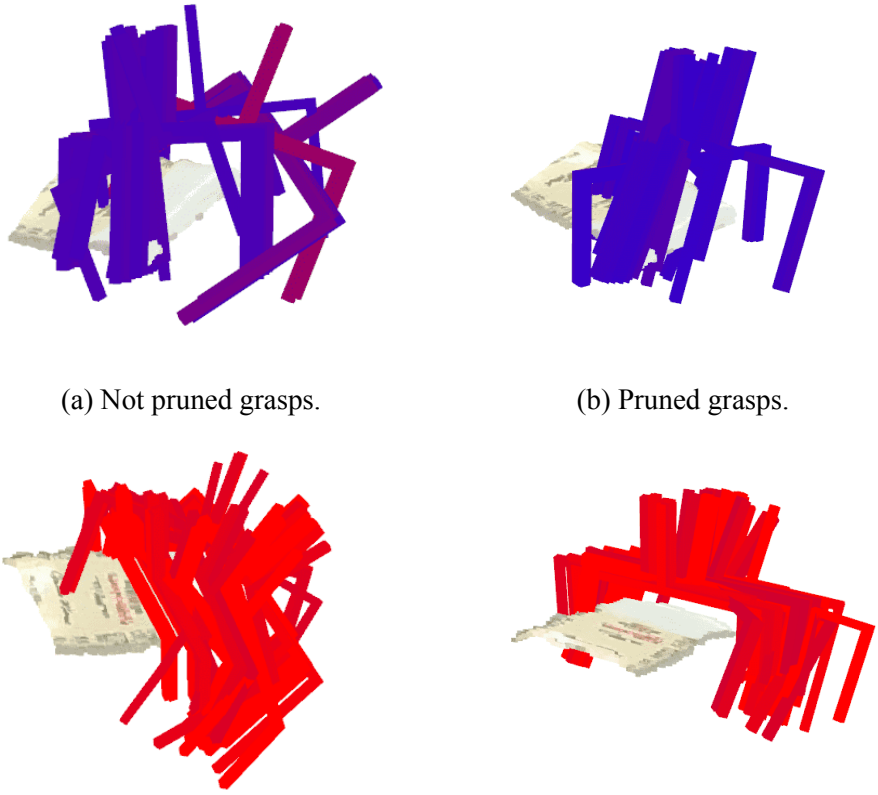
The evaluation is done considering all the segmentation and grasp detection configurations, resulting in 8 tables, which are shown in Appendix A. These tables show the metric results for each object and for each segmentation-grasp detection pair.

For the sake of clarity, only the results presented in Table 4.3 are analyzed. In particular, these are the results averaged over the objects for each models pair. It is evident that, in all configurations, a significant number of grasps are discarded, up to 90% of them, especially due to the exceedance of the roll and yaw ranges. Furthermore, the number of frames utilized for grasp generation remains consistent, with the exception of the combination *Complete-EgoHOS-6-DoF Graspnet*. This issue is probably caused by the presence of outliers in the point cloud, which makes the generation more challenging for that particular network.

Table 4.4 shows the distribution of grasps with respect to the hand and with respect to the best grasp. From the results, it is possible to notice that the D_{b_h} is generally higher for the set of pruned grasps than for the grasps generated by the model. This suggests that the grasp poses of the pruned set are concentrated on the opposite side of the object with respect to the hand position. This is confirmed by D_{b_g}, which tends to be smaller for the pruned set. Overall, the distances of the pruned grasps are approximately at 17 to 27 cm from the hand, while spanning from 15 to 23 cm for the not pruned set. Moreover, the distances between all grasps and the best one highlight that they tend to be closer on the pruned set.

For what concern the timing aspect, Table 4.5, shows the average time required by the networks to process the input and infer the grasps, along with the total time, which includes the time needed to process, infer and publish the result. The faster networks were *GraspNet* and *Contact-GraspNet*, which take 0.32 and 0.485 seconds per frame, respectively.

In light of the obtained results, especially those related to the timing aspect, the selected models for the online evaluation were *GraspNet* and *Contact-GraspNet*.



(a) Not pruned grasps.

(b) Pruned grasps.

(c) Not pruned grasps.

(d) Pruned grasps.

Figure 4.6: Difference between the generated grasps of GraspNet with the isolated object point cloud, Figures 4.6a and 4.6b, and the generated grasps with a virtual plane placed at distance 1 mm along the z-axis of the camera frame, Figures 4.6c and 4.6d. These images show the first 50 grasps sorted according to their score.

Configurations	Frames	Grasp quantity	Pruned grasps	% Discarded	Roll	Pitch	Yaw	Distance
GraspNet no plane (FE)	3757	4093952	3228870	78.8%	**1255283**	869767	509677	594143
GraspNet no plane (CE)	3606	2958387	2266137	76.6%	271948	224795	**915042**	854352
GraspNet plane 1mm (FE)	3446	2914718	2205757	75.6%	288371	230995	**944987**	741404
GraspNet plane 1mm (CE)	3751	4065901	3275822	80.5%	**1072547**	814174	599253	789848
6-DoF GraspNet no plane (FE)	3913	3368926	3036588	90.1%	**2067209**	442386	278727	248266
6-DoF GraspNet no plane (CE)	886	2806070	612669	21.8%	**451030**	48861	32320	80458
Contact-GraspNet no plane (FE)	3141	232001	142090	61.2%	25119	29015	43787	**44169**
Contact-GraspNet no plane (CE)	2994	111539	65685	58.8%	12628	16909	**18284**	17864

Table 4.3: Offline results obtained with all the configurations of the segmentation models, namely *Fast-EgoHOS* (FE) and *Complete-EgoHOS* (CE), with the grasp detection ones, namely *GraspNet*, *6-DoF GraspNet* and *Contact-GraspNet*. All results are reported in Appendix A.

Configurations	Pruned to Hand [m]	Not Pruned to Hand [m]	Pruned to Best Grasp [m]	Not Pruned to Best Grasp [m]
GraspNet no plane (FE)	0,1748	0,1607	**0,0327**	0,0417
GraspNet no plane (CE)	**0,1946**	0,1658	0,0467	0,0561
GraspNet plane 1mm (FE)	0,1730	0,1599	**0,0234**	0,0358
GraspNet plane 1mm (CE)	**0,1918**	0,1628	0,0241	0,0398
6-DoF GraspNet no plane (FE)	0,2078	0,1796	0,0661	0,1526
6-DoF GraspNet no plane (CE)	**0,2745**	0,2385	**0,0622**	0,1792
Contact-GraspNet no plane (FE)	0,2143	0,2009	**0,0346**	0,0675
Contact-GraspNet no plane (CE)	**0,2243**	0,2058	0,0431	0,0830

Table 4.4: Offline results expressing the mean distance between the best grasp to the hand (D_{b_h}) and the mean distance between the best grasp and all the other grasps of the same set (D_{b_g}). They are obtained with all the configurations of the segmentation models, namely *Fast-EgoHOS* (FE) and *Complete-EgoHOS* (CE), with the grasp detection ones, namely *GraspNet*, *6-DoF GraspNet* and *Contact-GraspNet*. All results are reported in Appendix A.

Models	Inference [s]	PC processing [s]	Total [s]
Graspnet	**0.24**	**0.03**	**0.319**
6-DoF GraspNet	4.54	0.04	5.55
Contact-GraspNet	0.41	0.06	0.485

Table 4.5: Average time needed for each of the grasp detection models.

4.3 Online Evaluation

The online evaluation consists of testing the transfer of all the 76 object-pose combinations, each for 3 times, resulting in a total of 228 handovers. Therefore, each object is tested 12 times.

As stated in [15], there is no a standard method for qualitatively evaluating the human-robot interaction, as the choice of the metrics highly depends on the task to be performed. One of the most frequently used metrics for H2R handovers is the success rate, which can be defined as the number of successful handovers divided by the total number of trials. During the experiments, an handover has been considered successful if it allows the robot to grasp the object and hold it firmly for a few seconds, without touching either the human hand or the object during the robot's motion and without pinching the hand.

Referring to this general validation approach, the handover operations have been evaluated considering two distinct metrics: the success rate, which is computed as the number of successful handovers over the total attempts, and the task completion time, which is the time needed to execute an handover in all its phases. In fact, referring to the handover stages in Section 4.1, it is possible to identify three main phases, not directly related to the segmentation or grasp detection modules. These are the selection of the best grasp, the trajectory planning and the actuation of it.

In light of the obtained results, hand-object segmentation method is *Fast-EgoHOS*, since a precise object mask is not needed for generating grasps and is suitable for a real-time approach. Instead, the chosen grasp generation models are *GraspNet*, with the virtual plane placed at distance 1 mm, and *Contact-GraspNet*. *6-DoF GraspNet* was not tested in a real environment because, in its current implementation, it is not optimized for a real-time approach.

Due to the presence of some unpredictable errors attributable to the motion planner, such as the robotic arm touching the object during its motion or the exceeding of the joint limits. The success rate has been evaluated according to two different protocols. One through a simple point-based assignment system, in which points are assigned according to the following criteria: 1 point, if the robot is able to grasp the object; 0.5 points, if the robot touches the object during its motion or if the motion planner aborts[2], provided that the robot could have successfully picked

[2]This is a problem caused by the exceeding of the joint limits, especially during the approaching phase, where the robot follows a straight trajectory before closing its end effector. This problem was mainly caused by the motion planner and was not managed by us.

up the object; 0 points, if it is unable to grasp the object, i.e. the grasp is not correct, or if no plans are found after three attempts or if the hand is touched.

The other is a binary assignment system which assigns 1 point for success and 0 for failure, without taking into account the issues related to the motion planner. Therefore, the handover was considered successful also if the motion planner's issues occurred.

Instead, for evaluating the efficiency of a handover, the task completion time metric was computed.

Tables 4.7 and 4.8 show the handover results with the *GraspNet* model for both point systems. Table 4.7 shows the results for the point-based system. The object with the highest mean success rate is the blue cube, with 0.916, while the one with the lowest value is the controller, with 0.416. Instead, with the binary point system, shown in Table 4.8, the objects with highest score are the blue cube, the hand cream and the wood stick, where every attempt was successful, and the one with lowest value is the controller, with half of the attempts wrong.

Tables 4.9 and 4.10 present the handover results with the *Contact-GraspNet* model. The object with the higher mean success rate obtained with the point-based system, see Table 4.7, is the wood stick, with 0.958, while the one with the lowest value is the hard disk box in vertical position, i.e. showing its shorter side to the camera, with 0.583. With the other assignment system, shown in Table 4.8, the objects with highest score are the marker, the spray can and the wood stick, where every attempt was considered successful, and the one with lowest value is the tape in the vertical position, i.e. not showing the hole to the camera, with 0.583.

The obtained results are highly dependent on the utilized point assignment system. If the issues of the motion planner were considered, then the overall score would be generally low, vice versa for the binary system. Between the two methods, the one with *Contact-Graspnet* performed better, achieving a higher total mean success rate with both point systems. Considering the binary point system, the combination *Fast-EgoHOS-Contact-Graspnet* reached a mean success rate of 0.829, while *Fast-EgoHOS-Graspnet* of 0.803.

Finally, Table 4.6 shows the results for the time metric, along with their mean success rate. These are comparable: each attempt takes approximately 20 seconds to be executed, 3 of which for selecting and planning the motion to the best grasp. The two main reasons for the prolonged execution time are, in part, intentional. Firstly, the robot speed is limited for safety reasons. Secondly, the robot does not immediately move towards the optimal grasp pose; instead, it moves to an approach position, from which it only needs to perform a straight motion along its approaching axis. This motion is the cause of the frequent abortions given by the planner.

Based on the experiments, it was observed that cylindrical objects and rectangular ones, with dimensions compatible with the end effector's maximum opening size, lead to a higher success rate. In particular, this was observed with the wood stick, the blue cube, and the remote.

Model	Selection [s]	Planning [s]	Attempts [s]	Mean Success Rate
GraspNet	0.553	**2.555**	**19.133**	0.803
Contact-GraspNet	**0.342**	3.01	21.655	**0.829**

Table 4.6: Results of the models in terms of average time needed to prune grasps and select which is the best one in terms of score, average time for the motion planner to find a plan, and average time for the attempt.

Conversely, objects with a particular shape, such as the controller, the tape and the hammer, or that require the end effector to be opened at maximum width, i.e. the white rectangle and the hard disk box, result in a lower success rate.

It has been observed that the generation of feasible grasps is challenging for objects whose point cloud is not precise enough. In particular, this can be observed with the tape, which has a point cloud that tends to be incomplete due to its reflective surface and thin size of the border.

Pose / Objects	|	—	\	/	Mean Success Rate (Point-based)
ball	s+a+t = 2	s+a+a = 2	s+s+t = 2.5	s+f+t = 1.5	0.667
blue_cube	s+t+t = 2	s+s+s = 3	s+s+s = 3	s+t+f = 3	**0.916**
controller	s+f+f = 1	s+t+f = 1.5	s+f+f = 1	s+t+f = 1.5	0.416
hammer	t+a+f = 1.5	s+f+f = 1	s+t+f = 1.5	s+a+f = 1.5	0.458
hand_cream	s+s+s = 3	s+s+t = 2.5	s+s+t = 2.5	s+s+t = 2.5	0.875
hard_disk_box_h	s+t+np = 1.5	s+f+f = 1	s+s+f = 2	s+t+f = 1.5	0.5
hard_disk_box_v	s+s+s = 3	s+t+f = 1.5	s+s+t = 2.5	s+s+s = 3	0.833
marker	s+s+s = 3	s+s+s = 3	s+t+f = 1.5	s+t+f = 1.5	0.75
pink_object	s+a+t = 2	s+s+t = 2.5	s+s+f = 2	s+a+t = 2	0.708
raspberry_box_h	s+s+f = 2	s+f+f = 1	s+t+f = 1.5	s+t+f = 1.5	0.5
raspberry_box_v	s+s+s = 3	s+s+a = 2.5	s+s+t = 2.5	t+t+f = 1	0.75
remote	s+s+a = 2.5	s+s+t = 2.5	s+s+f = 2	s+s+s = 3	0.833
screwdriver	s+s+s = 3	s+s+t = 2.5	s+s+f = 2	s+s+f = 2	0.791
spray	s+s+s = 3	s+t+f = 1.5	s+s+t = 2.5	s+s+s = 3	0.833
tape_h	s+s+f = 2	s+s+f = 2	s+s+f = 2	t+f+f = 0.5	0.541
tape_v	s+s+f = 2	s+f+f = 1	s+s+f = 2	s+t+f = 1.5	0.541
white_rectangle_h	s+t+t = 2	s+t+t = 2	s+t+f = 1.5	f+f+f = 0	0.458
white_rectangle_v	s+a+t = 2	s+a+f = 1.5	s+s+s = 3	a+a+t = 1.5	0.583
wood	s+s+t = 2.5	s+s+s = 3	s+a+t = 2	s+s+s = 3	0.875
Total					**0.679**

Table 4.7: Results obtained with *GraspNet*. Among the 228 handovers, the robot has touched (t) the object during its motion 39 times, has aborted (a) 13 times, has not found the plan (np) 1 time, has failed (f) the grasping procedure 45 times and has successfully (s) grasped the object 130 times. Results are represented as sum of three values, one for each attempt. The Mean Success Rate (Point-based) considers as points to be assigned 0, 0.5 and 1, depending on the case.

Objects \ Pose	|	—	\	/	Mean Success Rate (0/1)
ball	3	3	3	2	0.917
blue_cube	3	3	3	3	1
controller	1	2	1	2	0.5
hammer	2	1	2	2	0.583
hand_cream	3	3	3	3	1
hard_disk_box_h	3	1	2	2	0.667
hard_disk_box_v	3	2	3	3	0.917
marker	3	3	2	2	0.833
pink_object	3	3	2	3	0.917
raspberry_box_h	2	1	2	2	0.583
raspberry_box_v	3	3	3	2	0.917
remote	3	3	2	3	0.917
screwdriver	3	3	2	2	0.833
spray	3	2	3	3	0.917
tape_h	2	2	2	1	0.583
tape_v	2	1	2	2	0.583
white_rectangle_h	3	3	2	0	0.667
white_rectangle_v	3	2	3	3	0.917
wood	3	3	3	3	1
Total					**0.803**

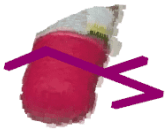
Table 4.8: Results obtained with *GraspNet*. Among the 228 handovers, the grasping procedure failed (f) 45 times and it has successfully (s) grasped the object 183 times. The Mean Success Rate (0/1) considers as points to be assigned 0 and 1.

Objects \ Pose	|	—	\	/	**Mean success rate (Point-based)**
ball	s+s+a = 2.5	s+s+f = 2	s+s+a = 2.5	s+s+s = 3	0.833
blue_cube	s+s+t = 2.5	s+s+t = 2.5	s+s+f = 2	s+s+a = 2.5	0.792
controller	s+s+np = 2	s+s+t = 2.5	s+s+f = 2	s+f+f = 1	0.458
hammer	s+np+f = 1	s+a+f = 1.5	s+s+a = 2.5	s+s+f = 2	0.583
hand_cream	s+s+t = 2.5	s+a+t = 2	s+s+np = 2	s+s+s = 3	0.792
hard_disk_box_h	s+s+s = 3	t+f+f = 0.5	s+s+t = 2.5	s+a+f = 1.5	0.625
hard_disk_box_v	s+t+f = 1.5	s+f+f = 1	s+s+a = 2.5	s+s+s = 3	0.458
marker	s+s+a = 2.5	s+s+s = 3	s+s+s = 3	s+s+a = 2.5	0.917
pink_object	s+s+f = 2	s+t+f = 1.5	s+a+f = 1.5	s+a+f = 1.5	0.542
raspberry_box_h	s+t+t = 2	s+s+s = 3	s+t+f = 1.5	s+s+f = 2	0.708
raspberry_box_v	s+s+s = 3	s+s+s = 3	s+t+f = 1.5	s+s+s = 3	0.875
remote	s+s+s = 3	s+s+t = 2.5	s+s+a = 2.5	s+s+s = 3	0.917
screwdriver	s+s+np = 2	s+a+t = 2	s+s+a = 2.5	s+s+f = 2	0.708
spray	s+s+s = 3	s+s+s = 3	s+s+t = 2.5	s+a+t = 2	0.875
tape_h	s+s+t = 2.5	s+s+a = 2.5	s+s+s = 3	s+f+f = 1	0.75
tape_v	s+s+f = 2	s+t+f = 1.5	s+f+f = 1	s+s+f = 2	0.542
white_rectangle_h	s+s+s = 3	s+s+np = 2	s+s+t = 2.5	s+s+f = 2	0.792
white_rectangle_v	s+s+s = 3	s+t+f = 1.5	s+a+t = 2	s+f+f = 1	0.625
wood	s+s+s = 3	s+s+a = 2.5	s+s+s = 3	s+s+s = 3	0.958
Total					**0.724**

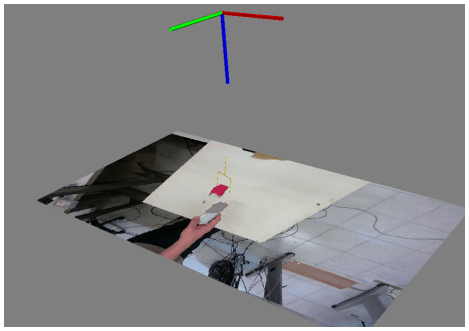
Table 4.9: Results obtained with *Contact-GraspNet*. Among the 228 handovers, the robot has touched (t) the object during its motion 21 times, has aborted (a) 19 times, has not found the plan (np) 5 times, has failed (f) the grasping procedure 33 times and has successfully (s) grasped the object 150 times. Results are represented as sum of three values, one for each attempt. The Mean Success Rate (Point-based) considers as points to be assigned 0, 0.5 and 1, depending on the case.

Objects \ Pose	|	—	\	/	Mean success rate (0/1)
ball	3	2	3	3	0.917
blue_cube	3	3	2	3	0.917
controller	2	3	2	1	0.667
hammer	1	2	3	2	0.667
hand_cream	3	3	2	3	0.917
hard_disk_box_h	3	1	3	2	0.75
hard_disk_box_v	2	1	3	3	0.75
marker	3	3	3	3	1
pink_object	2	2	2	2	0.667
raspberry_box_h	3	3	2	2	0.833
raspberry_box_v	3	3	2	3	0.917
remote	3	3	3	3	1
screwdriver	2	3	3	2	0.833
spray	3	3	3	3	1
tape_h	3	3	3	1	0.833
tape_v	2	2	1	2	0.583
white_rectangle_h	3	2	3	1	0.75
white_rectangle_v	3	2	3	1	0.75
wood	3	3	3	3	1
Total					**0.829**

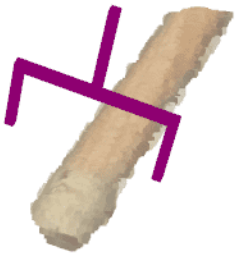
Table 4.10: Results obtained with *Contact-GraspNet*. Among the 228 handovers, the grasping procedure failed (f) 38 times and it has successfully (s) grasped the object 189 times. The Mean Success Rate (0/1) considers as points to be assigned 0 and 1.



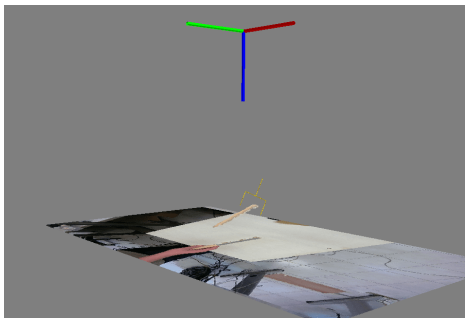
(a) Spray can with GraspNet.



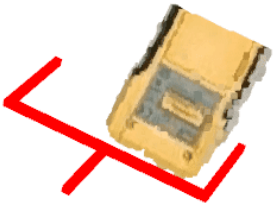
(b) Spray can with Contact-GraspNet.



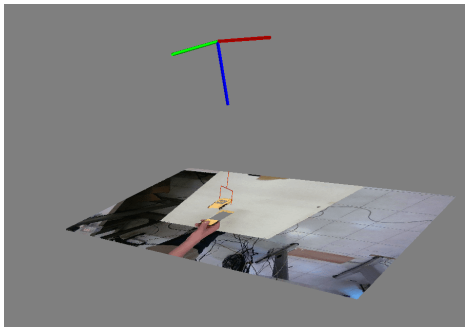
(c) Wood stick with GraspNet.



(d) Wood stick with Contact-GraspNet.



(e) Remote with GraspNet.



(f) Remote with Contact-GraspNet.

Figure 4.7: Example of the generated best grasp on the objects belonging to the dataset. Other examples are shown in Appendix B

4.4 Problems and Limitations

During the experimental phase, some issues were observed. In particular, there were difficulties when segmenting certain objects, namely the hammer, the tape and the marker, mainly due to their reflective surface. Moreover, the segmentation process can also be problematic for

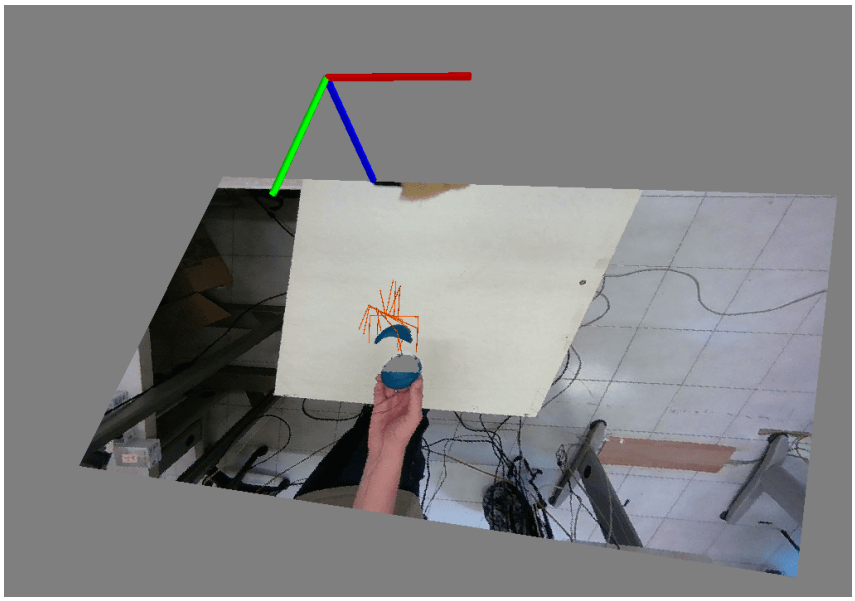
the *Fast-EgoHOS* method, especially when dealing with transparent objects. Another issue was the inability to find the motion plan. This problem occurred especially when the robot had to perform the straight motion from the approach position. Finally, some problems were observed when generating grasps. These were more evident for thick objects, which only allow grasps at maximum grasp width, such as the hand cream, the hammer, the white rectangle and the controller, and thin objects, such as the screwdriver, due to their limited number of points in the point cloud.

The selection of the hand-grasp distance threshold and the roll, pitch and yaw intervals were determinant for choosing which of the generated grasps to retain, with the objective of having them placed on the same side of the robot and above the object. In some cases, these ranges were too strict, as can be observed in Figures 4.8a and 4.8b, where all the generated grasps were discarded due to the applied criteria. In this specific case, roll, pitch and the distance criteria discarded one grasp each, while the yaw discarded two of them. The case in which all grasps are discarded may happen for object whose point cloud has low resolution or for thick objects.

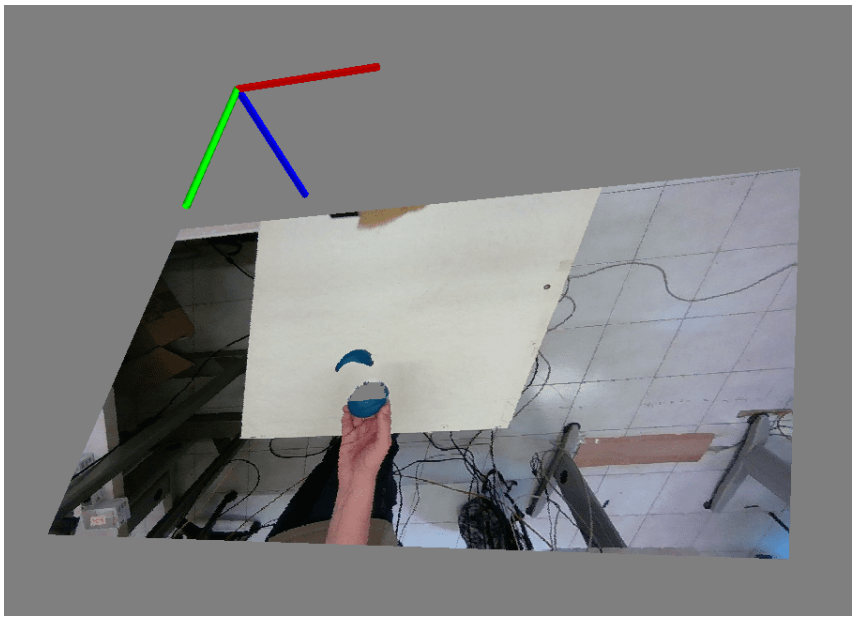
These aforementioned problems led the model to generate a limited number of suitable grasps or, alternatively, discard them all. During the online evaluation, this was not a critical issue, since the cases in which the models were unable to generate at least one grasp were not so frequent enough to slow down the overall process.

The most frequently observed issues, which were not directly related to the implementation of the modules or to the type of objects, were attributed to the motion planner. The main ones were related to the planner's inability to find a trajectory for reaching the grasp pose, the abortion during its motion, and the robot's contact with the object. The latter issue was largely due to the object not being treated as a collision box in the scene, which resulted in the planner lacking knowledge about its presence.

Finally, the decision to employ an open-loop approach, that is without dealing with potential hand motion during the robot's movements, rather than a closed-loop one, was primarily influenced by the limitations of the depth sensor. In fact, the depth camera has a minimum depth distance of approximately 30 cm, which results in some measuring errors when the end effector is in close proximity to the object.



(a)



(b)

Figure 4.8: Issue that occurred when the roll, pitch and yaw ranges are narrow, see Section 3.3. Figure 4.8a shows the generated grasps from the Contact-GraspNet model, while 4.8b displays the remaining ones after the pruning operation. In this case no grasps are left.

Chapter 5

Conclusions and Future Works

This thesis presented an approach to the H2R handover task for generic objects based on a pipeline consisting of three phases. These stages are: a segmentation module to obtain a hand-object mask of the scene, which is observed from an egocentric point of view with respect to the robot; a grasp generation module to generate a set of grasps, starting from the hand-object mask and the point cloud of the object; and finally, a grasp selection module to select which of them are suitable for execution.

In particular, since the H2R task requires the system to be accurate and fast at the same time, different state-of-the-art approaches for each module have been evaluated. For the segmentation module, whose objective is to distinguish between the hand and the object being held, the compared methodologies were *Fast-EgoHOS* and *Complete-EgoHOS*, both based on the EgoHOS [1] model. Between these two choices, *Fast-EgoHOS* was selected as it offered a good balance in terms of accuracy-time trade-off. With regard to the grasp generation module, an evaluation has been conducted on the following state-of-the-art models: GraspNet [2], 6-DoF GraspNet [3], and Contact-GraspNet [4]. The evaluation objectives were: firstly, the models' capacity to generate a dense set of feasible grasps on the given object point cloud in a short amount of time; and secondly, to verify the possibility of adapting these models trained on static objects for the H2R handover task. In order to select which of the grasp poses were feasible for execution, three discarding criteria have been chosen based on the grasp distance to the hand, its distance to the object, and its orientation. Among the three model proposals, GraspNet and Contact-GraspNet were selected because they offer a good balance between grasp generation and total execution time.

Each module has been subjected to both offline and online evaluation. The offline evaluation was mainly used to identify which combination of segmentation-grasp detection models performed better on a recorded dataset. With regard to the segmentation, the evaluation was performed in terms of Intersection Over Union and Average Execution Time. Instead, for the

grasp detection, the evaluation was conducted in terms of remaining grasps, after the application of the chosen pruning criteria, and of distance-based metrics, with the objective of verifying how grasps are distributed with respect to the hand. Based on the obtained offline results, the segmentation-grasp generation pairs selected for the online evaluation were *Fast-EgoHOS-GraspNet* and *Fast-EgoHOS-Contact-GraspNet*.

The online evaluation was performed on a set of 15 distinct objects, for a total of 228 handovers. Each object was presented in four different orientations to test the robustness of the system to the way objects can be handled. In this case, for evaluating the selected pair of models, two metrics were considered: the success rate, which is up 82.9% with the *Fast-EgoHOS-Contact-GraspNet* pair, and the time required for each grasping attempt, which takes approximately 20 seconds, 3 of which for selecting and planning the motion to the best grasp.

From the results, it has been observed that adapting a grasp detection module from a static to a dynamic scene is feasible, but with a trade-off in terms of accuracy, total number of grasps and the execution time. Grasps are distributed on the entire object point cloud and the best one is averagely positioned far from the hand. These models are capable of generating a dense set of grasps on the object point cloud, however only a subset of them are suitable for execution, since the correct execution highly depends on the robot's physical limits and on the applied safety policies.

In light of the limitations described in Section 4.4, some future work may focus on implementing improvements to the system to enhance its reliability and speed, as well as defining common evaluation criteria to ensure a fair comparison with other implementations. To improve the system's reliability, collision boxes can be applied on every obstacle in the scene, including the hand, the person, and the object itself. This would enable the motion planner to compute a trajectory that avoids these obstacles during the motion. Moreover, the overall execution time can be reduced if the robot's followed optimal paths. This issue could be reduced implementing a custom trajectory planner.

Finally, the implementation of a multi-camera system allows the approach to be closed-loop, thereby overcoming the limitations caused by the depth sensor when objects are under a certain distance. The presence of an *eye-to-hand* camera, which is not mounted on the end effector, can enhance the robot's perception capabilities, enabling it to perceive the scene from multiple point of views, helping in case of occlusions.

In conclusion, even after applying the criteria to force the robot to pick up the object in a certain way, the implemented system is not free from error and cannot be used without supervision. Further development is necessary to improve the performance and to make it more stable, reliable and fast.

Appendices

Appendix A

Offline Evaluation

The following tables show the complete results obtained with all the configurations of the segmentation models, namely *Fast-EgoHOS* (FE) and *Complete-EgoHOS* (CE), with the grasp detection ones, namely *GraspNet*, *6-DoF GraspNet* and *Contact-GraspNet*.

The results are in Tables: A.1 and A.2 for what concern the ones obtained with GraspNet without the virtual plane; A.3 and A.4 for GraspNet with the virtual plane placed at 1 mm; A.5 and A.6 for 6-DoF GraspNet model; A.7 and A.8 for the results with Contact-GraspNet.

Object	Frames	Grasp quantity	Pruned grasps	% Discarded	Roll	Pitch	Yaw	Distance
ball	163	216064	154000	71.3%	63905	65663	16780	7652
blue_cube	211	216064	186123	86.1%	62836	92595	19801	10891
controller	213	218112	165531	75.9%	33779	36910	41455	53387
hammer	210	215040	169157	78.7%	50238	26876	53067	38976
hand_cream	209	214016	172988	80.8%	31018	42216	39788	59966
hard_disk_box_h	210	215040	153123	71.2%	46939	16511	15484	74189
hard_disk_box_h	211	219136	188424	85.9%	85249	52069	14460	36646
marker	201	214016	189667	88.6%	111666	51121	8189	18691
pink_object	199	212992	184212	86.5%	75105	81839	17436	9832
raspberry_box_h	213	218112	164719	75.5%	40203	23505	46639	54372
raspberry_box_v	212	217088	177521	81.8%	70227	19990	37810	49494
remote	206	214016	181547	84.8%	54899	66125	25731	34792
screwdriver	75	216064	59375	27.5%	26759	7348	711	24557
spray	211	216064	180663	83.6%	49693	56990	28538	45442
tape_h	211	217088	172989	79.7%	142383	17057	6432	7117
tape_v	180	215040	166208	77.3%	74466	57448	26869	7425
white_rectangle_h	207	211968	188631	89.0%	60458	37621	70574	19978
white_rectangle_v	204	211968	182340	86.0%	58337	81635	26874	15494
wood	211	216064	191652	88.7%	117123	36248	13039	25242
Total:	3757	4093952	3228870	78.8%	1255283	869767	509677	594143

Table A.1: Offline results obtained with the configuration *Fast-EgoHOS-GraspNet* with input the point cloud of the object, without virtual plane.

Object	Frames	Grasp quantity	Pruned grasps	% discarded	Roll	Pitch	Yaw	Distance
ball	157	25849	20014	77.4%	6721	9495	1576	2222
blue_cube	167	32161	22937	71.3%	4393	1970	5433	11141
controller	150	54164	41020	75.7%	9991	7282	10326	13421
hammer	139	109512	81838	74.7%	19566	5431	37219	19622
hand_cream	206	174517	158020	90.5%	19272	30079	88409	20260
hard_disk_box_h	162	63402	52370	82.6%	12842	8298	10584	20646
hard_disk_box_h	200	216064	190290	88.0%	12269	21010	127352	29659
marker	172	105783	69043	65.2%	10217	6140	20265	32421
pink_object	205	122141	86047	70.4%	6199	6636	40514	32698
raspberry_box_h	204	209919	188470	89.8%	20978	6847	124300	36345
raspberry_box_v	212	216835	180670	83.3%	42657	31693	67708	38612
remote	198	212992	185922	87.3%	16606	6497	113845	48974
screwdriver	206	211967	153097	72.2%	10855	12072	75027	55143
spray	208	170120	121312	71.3%	11536	8466	44723	56587
tape_h	205	181151	132790	73.3%	9191	15966	46880	60753
tape_v	210	207714	155131	74.6%	10406	19591	53372	71762
white_rectangle_h	206	216064	129971	60.1%	4041	11127	27475	87328
white_rectangle_v	201	212992	137780	64.6%	18504	4537	13120	101619
wood	198	215040	159415	74.1%	25704	11658	6914	115139
Total:	3606	2958387	2266137	76.6%	271948	224795	915042	854352

Table A.2: Offline results obtained with the configuration *Complete-EgoHOS-GraspNet* with a plane placed at distance 1 mm along the z-axis of the camera frame.

Object	Frames	Grasp quantity	Pruned grasps	% Discarded	Roll	Pitch	Yaw	Distance
ball	200	216064	184732	85.5%	9440	25871	117679	31742
blue_cube	210	216639	174456	80.5%	42585	26718	69300	35853
controller	118	18652	13507	72.4%	4826	6125	456	2100
hammer	195	152496	129586	84.9%	22686	28131	57053	21716
hand_cream	209	120753	87195	72.2%	8918	8183	41306	28788
hard_disk_box_h	100	21793	13695	62.8%	1845	850	4802	6198
hard_disk_box_h	207	168556	115717	68.6%	6711	7311	39957	61738
marker	207	214016	138684	64.8%	19471	5678	19377	94158
pink_object	207	212992	187108	87.8%	27860	9081	116008	34159
raspberry_box_h	145	44867	34608	77.1%	8272	7590	6856	11890
raspberry_box_v	108	46764	33480	71.5%	6616	4764	9943	12157
remote	208	182429	141265	77.4%	17682	15776	49702	58105
screwdriver	204	216064	139416	64.5%	28385	5935	10894	94202
spray	210	204145	146468	71.7%	17454	22337	51116	55561
tape_h	212	213119	179104	84.0%	30713	20984	86692	40715
tape_v	209	215040	183247	85.2%	18724	10318	136171	18034
white_rectangle_h	79	22298	15082	67.6%	1840	2490	6395	4357
white_rectangle_v	207	211967	150055	70.8%	10430	13045	84289	42291
wood	211	216064	138352	64.0%	3913	9808	36991	87640
Total:	3446	2914718	2205757	75.6%	288371	230995	944987	741404

Table A.3: Offline results obtained with the configuration *Fast-EgoHOS-GraspNet* with a plane placed at distance 1 mm along the z-axis of the camera frame.

Object	Frames	Grasp quantity	Pruned grasps	Percentage discarded	Roll	Pitch	Yaw	Distance
ball	167	216064	161087	74.5%	60362	62340	22177	16208
blue_cube	212	217088	177887	81.9%	50413	84945	22854	19675
controller	213	218112	168175	77.1%	30943	35405	39964	61863
hammer	210	214685	169323	78.8%	44364	28425	51156	45378
hand_cream	209	214016	188203	87.9%	24499	40185	42542	80977
hard_disk_box_h	210	215040	156239	72.6%	41260	14666	18373	81940
hard_disk_box_h	214	219136	197596	90.1%	68121	51234	38326	39915
marker	169	212992	156069	73.2%	86131	35713	9033	25192
pink_object	179	212992	170394	80.0%	66700	67689	18435	17570
raspberry_box_h	213	218112	169219	77.5%	44179	19194	48279	57567
raspberry_box_v	211	217088	171696	79.0%	61062	13781	36340	60513
remote	198	214016	178251	83.2%	41756	47499	39582	49414
screwdriver	150	215029	140609	65.3%	83210	22879	2612	31908
spray	211	216064	186892	86.5%	38762	56877	34311	56942
tape_h	179	194501	153170	78.7%	49418	27043	44450	32259
tape_v	184	210966	172659	81.8%	62686	51035	35228	23710
white_rectangle_h	204	211968	174258	82.2%	43956	38396	51485	40421
white_rectangle_v	207	211968	190328	89.8%	56955	77626	31331	24416
wood	211	216064	193767	89.6%	117770	39242	12775	23980
Total:	3751	4065901	3275822	80.5%	1072547	814174	599253	789848

Table A.4: Offline results obtained with the configuration *Complete-EgoHOS-GraspNet* with a plane placed at distance 1 mm along the z-axis of the camera frame.

Object	Frames	Grasp quantity	Pruned grasps	% Discarded	Roll	Pitch	Yaw	Distance
ball	209	141528	128909	91.0%	91349	12540	20115	4905
blue_cube	210	158757	142291	89.6%	95380	13380	23714	9817
controller	212	180021	160002	88.8%	115559	29593	5171	9679
hammer	205	193135	175364	90.7%	111892	35908	8222	19342
hand_cream	209	178741	161382	90.2%	102150	26545	11316	21371
hard_disk_box_h	210	194759	179614	92.2%	139694	23066	8889	7965
hard_disk_box_h	213	195447	170683	87.3%	130061	14425	15222	10975
marker	200	187254	165319	88.2%	107813	25687	10408	21411
pink_object	205	117769	109871	93.2%	76192	8610	18355	6714
raspberry_box_h	213	198577	182258	91.7%	122550	33999	13530	12179
raspberry_box_v	212	188936	175658	92.9%	117496	33523	13540	11099
remote	205	153573	138760	90.3%	91341	17677	15110	14632
screwdriver	207	233233	209262	89.7%	129951	42355	9361	27595
spray	211	193444	173578	89.7%	111669	27378	16719	17812
tape_h	167	140629	116072	82.5%	82029	7519	18393	8131
tape_v	207	147311	133147	90.3%	95245	11514	20300	6088
white_rectangle_h	207	163187	150007	91.9%	100717	26832	16694	5764
white_rectangle_v	200	159679	141338	88.5%	97926	14953	22774	5685
wood	211	242946	223073	91.8%	148195	36882	10894	27102
Total:	3913	3368926	3036588	90.1%	2067209	442386	278727	248266

Table A.5: Offline results obtained with the configuration *Fast-EgoHOS-6-DoF GraspNet* with input the point cloud of the object, without virtual plane.

Object	Frames	Grasp quantity	Pruned grasps	Percentage discarded	Roll	Pitch	Yaw	Distance
ball	8	110575	5589	5.0%	4174	376	812	227
blue_cube	40	164573	27289	16.5%	17698	3103	5059	1429
controller	29	136023	19329	14.2%	14027	2224	1566	1512
hammer	107	126407	59108	46.7%	48352	2180	819	7757
hand_cream	5	148386	3517	2.3%	2535	343	126	513
hard_disk_box_h	37	149002	28474	19.1%	23069	1777	1862	1766
hard_disk_box_h	48	200983	35386	17.6%	25013	5321	611	4441
marker	78	147593	58038	39.3%	44372	3437	1568	8661
pink_object	9	129610	6987	5.3%	5534	434	704	315
raspberry_box_h	11	128920	8531	6.6%	6304	695	532	1000
raspberry_box_v	30	131602	19683	14.9%	15851	1303	716	1813
remote	5	133868	2894	2.1%	2260	77	87	470
screwdriver	134	148479	89875	60.5%	61549	4263	3352	20711
spray	19	170707	15324	8.9%	12240	508	433	2143
tape_h	23	151150	18894	12.5%	14337	510	1186	2861
tape_v	7	149393	4585	3.0%	3952	47	251	335
white_rectangle_h	129	156657	94415	60.2%	65544	16410	8610	3851
white_rectangle_v	9	167683	6680	3.9%	5412	279	696	293
wood	158	154459	108071	69.9%	78807	5574	3330	20360
Total:	886	2806070	612669	21.8%	451030	48861	32320	80458

Table A.6: Offline results obtained with the configuration *Complete-EgoHOS-6-DoF GraspNet* with input the point cloud of the object, without virtual plane.

Object	Frames	Grasp quantity	Pruned grasps	% Discarded	Roll	Pitch	Yaw	Distance
ball	164	5552	2981	53.7%	224	13	1111	1633
blue_cube	211	14896	9743	65.4%	1220	418	2772	5333
controller	182	4405	2522	57.2%	1	209	329	1983
hammer	151	11375	7016	61.7%	67	783	3529	2637
hand_cream	127	6721	3230	48.1%	5	1803	740	682
hard_disk_box_h	163	33060	24730	74.8%	9077	10968	3543	1142
hard_disk_box_h	163	22646	13791	60.9%	2254	3312	5642	2583
marker	164	5791	3490	60.3%	53	0	1543	1894
pink_object	177	5780	3948	68.3%	8	1	1912	2027
raspberry_box_h	106	16181	9078	56.1%	1757	3761	2882	678
raspberry_box_v	78	22370	7482	33.4%	1663	1740	3278	801
remote	161	5435	2737	50.3%	0	0	1380	1357
screwdriver	131	3968	1985	50.0%	40	0	734	1211
spray	205	9931	6321	63.6%	0	1	2563	3757
tape_h	206	9190	6505	70.8%	235	4	1639	4627
tape_v	209	10819	7477	69.1%	139	56	2787	4495
white_rectangle_h	128	17450	11583	66.4%	3936	4732	2534	381
white_rectangle_v	204	13465	8919	66.2%	3731	932	2506	1750
wood	211	12966	8552	65.9%	709	282	2363	5198
Total:	3141	232001	142090	61.2%	25119	29015	43787	44169

Table A.7: Offline results obtained with the configuration *Fast-EgoHOS-Contact-GraspNet* with input the point cloud of the object, without virtual plane.

Object	Frames	Grasp quantity	Pruned grasps	Percentage discarded	Roll	Pitch	Yaw	Distance
ball	185	1960	1085	55.3%	15	11	429	630
blue_cube	210	6106	4313	70.6%	565	357	1559	1832
controller	158	2532	1598	63.1%	3	186	617	792
hammer	149	5017	2970	59.1%	36	234	1569	1131
hand_cream	89	1500	921	61.4%	37	454	271	159
hard_disk_box_h	179	24442	18650	76.3%	6857	8172	2278	1343
hard_disk_box_h	176	8986	4325	48.1%	407	894	1747	1277
marker	157	2257	1310	58.0%	15	0	518	777
pink_object	156	1503	923	61.4%	13	1	421	488
raspberry_box_h	114	5569	2838	50.9%	742	956	813	327
raspberry_box_v	90	8796	3238	36.8%	983	1186	843	226
remote	163	1748	1016	58.1%	0	2	446	568
screwdriver	116	2206	1085	49.1%	82	2	310	691
spray	207	3116	1930	61.9%	2	1	606	1321
tape_h	151	4256	2729	64.1%	509	265	1272	683
tape_v	198	3120	2110	67.6%	35	56	734	1285
white_rectangle_h	115	17285	7653	44.2%	1713	3933	1519	488
white_rectangle_v	170	4033	2308	57.2%	336	94	905	973
wood	211	7107	4683	65.8%	278	105	1427	2873
Total:	**2994**	**111539**	**65685**	**58.8%**	**12628**	**16909**	**18284**	**17864**

Table A.8: Offline results obtained with the configuration *Complete-EgoHOS-Contact GraspNet* with input the point cloud of the object, without virtual plane.

Object	Pruned to Hand [m]	Not Pruned to Hand [m]	Pruned to Best Grasp [m]	Not Pruned to Best Grasp [m]
ball	0,1825	0,1563	0,0122	0,0287
blue_cube	0,1650	0,1454	0,0210	0,0327
controller	0,1904	0,1644	0,0428	0,0575
hammer	0,1986	0,1938	0,0380	0,0537
hand_cream	0,1691	0,1615	0,0352	0,0357
hard_disk_box_h	0,1846	0,1727	0,0530	0,0704
hard_disk_box_h	0,1862	0,1686	0,0284	0,0416
marker	0,1761	0,1716	0,0243	0,0341
pink	0,1672	0,1483	0,0109	0,0232
raspberry_box_h	0,1629	0,1508	0,0394	0,0519
raspberry_box_v	0,1803	0,1690	0,0364	0,0503
remote	0,1666	0,1558	0,0277	0,0325
screw	0,2666	0,2571	0,0353	0,0546
spray	0,1659	0,1502	0,0322	0,0356
tape_h	0,1458	0,1367	0,0168	0,0288
tape_v	0,1411	0,1253	0,0133	0,0273
white_rectangle_h	0,1558	0,1399	0,0330	0,0446
white_rectangle_v	0,1543	0,1379	0,0190	0,0298
wood	0,2169	0,2035	0,0390	0,0598
Total:	**0,1748**	**0,1607**	**0,0327**	**0,0417**

Table A.9: Results obtained with the *Fast-EgoHOS*-**GraspNet** model, without virtual plane, of the mean distance between the best grasp to the hand and the mean distance between the best grasp and all the other grasps of the same set.

Object	Pruned to Hand [m]	Not Pruned to Hand [m]	Pruned to Best Grasp [m]	Not Pruned to Best Grasp [m]
ball	0,1914	0,1533	0,0111	0,0366
blue_cube	0,1636	0,1447	0,0207	0,0356
controller	0,1942	0,1649	0,0426	0,0607
hammer	0,2014	0,1959	0,0482	0,0634
hand_cream	0,1980	0,1711	0,0844	0,0609
hard_disk_box_h	0,1878	0,1774	0,0541	0,0739
hard_disk_box_h	0,1971	0,1718	0,0375	0,0470
marker	0,1823	0,1762	0,0260	0,0418
pink	0,1857	0,1580	0,0096	0,0342
raspberry_box_h	0,1732	0,1622	0,0462	0,0759
raspberry_box_v	0,1867	0,1701	0,0479	0,0532
remote	0,1902	0,1596	0,0469	0,0371
screw	0,2412	0,2294	0,0393	0,1043
spray	0,1780	0,1505	0,0547	0,0422
tape_h	0,2765	0,1549	0,0698	0,1092
tape_v	0,1989	0,1377	0,0326	0,0531
white_rectangle_h	0,1694	0,1352	0,0480	0,0466
white_rectangle_v	0,1839	0,1381	0,0806	0,0382
wood	0,2187	0,2091	0,0310	0,0624
Total:	**0,1946**	**0,1658**	**0,0467**	**0,0561**

Table A.10: Results obtained with the *Complete-EgoHOS*-**GraspNet** model, without virtual plane, of the mean distance between the best grasp to the hand and the mean distance between the best grasp and all the other grasps of the same set.

Object	Pruned to Hand [m]	Not Pruned to Hand [m]	Pruned to Best Grasp [m]	Not Pruned to Best Grasp [m]
ball	0,1537	0,1336	0,0090	0,0264
blue_cube	0,1669	0,1532	0,0187	0,0324
controller	0,2101	0,1895	0,0145	0,0586
hammer	0,2005	0,1911	0,0279	0,0439
hand_cream	0,1802	0,1709	0,0289	0,0450
hard_disk_box_h	0,2040	0,1943	0,0261	0,0381
hard_disk_box_h	0,1846	0,1767	0,0340	0,0459
marker	0,1738	0,1580	0,0167	0,0312
pink	0,1716	0,1570	0,0108	0,0230
raspberry_box_h	0,1508	0,1402	0,0353	0,0429
raspberry_box_v	0,1647	0,1469	0,0284	0,0377
remote	0,1657	0,1523	0,0257	0,0315
screw	0,1941	0,1799	0,0207	0,0438
spray	0,1664	0,1482	0,0240	0,0376
tape_h	0,1467	0,1381	0,0137	0,0280
tape_v	0,1310	0,1206	0,0215	0,0282
white_rectangle_h	0,2144	0,2053	0,0116	0,0285
white_rectangle_v	0,1484	0,1414	0,0221	0,0319
wood	0,2090	0,1884	0,0346	0,0568
Total:	**0,1730**	**0,1599**	**0,0234**	**0,0358**

Table A.11: Results obtained with the *Complete-EgoHOS*-**GraspNet** model with the virtual plane placed at 1 mm. Results are about the mean distance between the best grasp to the hand and the mean distance between the best grasp and all the other grasps of the same set.

Object	Pruned to Hand [m]	Not Pruned to Hand [m]	Pruned to Best Grasp [m]	Not Pruned to Best Grasp [m]
ball	0,1585	0,1386	0,0101	0,0256
blue_cube	0,1694	0,1557	0,0223	0,0329
controller	0,2821	0,1738	0,0373	0,0714
hammer	0,2162	0,1916	0,0270	0,0526
hand_cream	0,1991	0,1744	0,0308	0,0472
hard_disk_box_h	0,2693	0,1943	0,0313	0,0522
hard_disk_box_h	0,1926	0,1797	0,0315	0,0452
marker	0,1780	0,1619	0,0146	0,0317
pink	0,1743	0,1615	0,0099	0,0222
raspberry_box_h	0,1918	0,1536	0,0347	0,0553
raspberry_box_v	0,2196	0,1721	0,0303	0,0449
remote	0,1757	0,1567	0,0229	0,0354
screw	0,1933	0,1791	0,0167	0,0465
spray	0,1704	0,1509	0,0255	0,0371
tape_h	0,1884	0,1601	0,0211	0,0717
tape_v	0,1652	0,1160	0,0216	0,0310
white_rectangle_h	0,1772	0,1475	0,0299	0,0413
white_rectangle_v	0,1530	0,1419	0,0213	0,0344
wood	0,2099	0,1887	0,0325	0,0557
Total:	**0,1918**	**0,1628**	**0,0241**	**0,0398**

Table A.12: Results obtained with the *Complete-EgoHOS*-**GraspNet** model with the virtual plane placed at 1 mm. Results are about the mean distance between the best grasp to the hand and the mean distance between the best grasp and all the other grasps of the same set.

Object	Pruned to Hand [m]	Not Pruned to Hand [m]	Pruned to Best Grasp [m]	Not Pruned to Best Grasp [m]
ball	0,1908	0,1502	0,0644	0,1365
blue_cube	0,1959	0,1642	0,0668	0,1376
controller	0,2231	0,2200	0,0740	0,1778
hammer	0,2170	0,1992	0,0782	0,1621
hand_cream	0,1964	0,1806	0,0574	0,1455
hard_disk_box_h	0,2463	0,2130	0,0861	0,1888
hard_disk_box_h	0,2269	0,1990	0,0666	0,1492
marker	0,2126	0,1820	0,0612	0,1432
pink	0,2064	0,1771	0,0454	0,1364
raspberry_box_h	0,1941	0,1595	0,0684	0,1664
raspberry_box_v	0,2098	0,1815	0,0687	0,1681
remote	0,2000	0,1748	0,0572	0,1426
screw	0,2308	0,2082	0,0603	0,1486
spray	0,1994	0,1676	0,0621	0,1446
tape_h	0,1910	0,1562	0,0473	0,1420
tape_v	0,1769	0,1466	0,0589	0,1348
white_rectangle_h	0,1887	0,1593	0,0775	0,1649
white_rectangle_v	0,1955	0,1604	0,0694	0,1374
wood	0,2419	0,2072	0,0640	0,1502
Total:	**0,2078**	**0,1796**	**0,0661**	**0,1526**

Table A.13: Results obtained with the *Fast-EgoHOS*-**6-DoF GraspNet** configuration of the mean distance between the best grasp to the hand and the mean distance between the best grasp and all the other grasps of the same set.

Object	Pruned to Hand [m]	Not Pruned to Hand [m]	Pruned to Best Grasp [m]	Not Pruned to Best Grasp [m]
ball	0,2896	0,2488	0,0513	0,1512
blue_cube	0,1778	0,1427	0,0605	0,1348
controller	0,2679	0,2703	0,0645	0,2054
hammer	0,3270	0,2830	0,0414	0,2097
hand_cream	0,2488	0,2116	0,0465	0,1855
hard_disk_box_h	0,2297	0,2046	0,0805	0,1626
hard_disk_box_h	0,2833	0,2494	0,0676	0,1525
marker	0,2928	0,2621	0,0472	0,1716
pink	0,3080	0,2822	0,0341	0,1226
raspberry_box_h	0,2834	0,2662	0,0570	0,1601
raspberry_box_v	0,2742	0,2742	0,0487	0,1825
remote	0,2803	0,2319	0,0841	0,2093
screw	0,3174	0,2668	0,0523	0,2020
spray	0,2844	0,2653	0,0378	0,1856
tape_h	0,1782	0,1508	0,0403	0,1565
tape_v	0,2807	0,2560	0,0420	0,1660
white_rectangle_h	0,1832	0,1608	0,0824	0,1629
white_rectangle_v	0,2496	0,1946	0,0579	0,1396
wood	0,3129	0,2599	0,0502	0,1943
Total:	**0,2745**	**0,2385**	**0,0622**	**0,1792**

Table A.14: Results obtained with the *Complete-EgoHOS*-**6-DoF GraspNet** configuration of the mean distance between the best grasp to the hand and the mean distance between the best grasp and all the other grasps of the same set.

Object	Pruned to Hand [m]	Not Pruned to Hand [m]	Pruned to Best Grasp [m]	Not Pruned to Best Grasp [m]
ball	0,2097	0,2053	0,0160	0,0274
blue_cube	0,1999	0,1893	0,0390	0,0526
controller	0,2098	0,1895	0,0451	0,0539
hammer	0,2971	0,2595	0,0439	0,0892
hand_cream	0,2093	0,2029	0,0170	0,0488
hard_disk_box_h	0,2352	0,2314	0,0579	0,1048
hard_disk_box_h	0,2362	0,2265	0,0397	0,0709
marker	0,2188	0,2120	0,0214	0,0434
pink	0,1943	0,1837	0,0212	0,0316
raspberry_box_h	0,1748	0,1629	0,0262	0,0779
raspberry_box_v	0,2895	0,2825	0,0414	0,0764
remote	0,1827	0,1713	0,0219	0,0288
screw	0,2628	0,2312	0,0291	0,0658
spray	0,2090	0,1954	0,0302	0,0447
tape_h	0,1598	0,1520	0,0239	0,0319
tape_v	0,1787	0,1677	0,0263	0,0354
white_rectangle_h	0,2088	0,1986	0,0307	0,0955
white_rectangle_v	0,2017	0,1911	0,0471	0,0757
wood	0,2547	0,2275	0,0347	0,0687
Total:	**0,2143**	**0,2009**	**0,0346**	**0,0675**

Table A.15: Results obtained with the *Fast-EgoHOS*-**Contact-GraspNet** configuration of the mean distance between the best grasp to the hand and the mean distance between the best grasp and all the other grasps of the same set.

Object	Pruned to Hand [m]	Not Pruned to Hand [m]	Pruned to Best Grasp [m]	Not Pruned to Best Grasp [m]
ball	0,2077	0,2020	0,0153	0,0283
blue_cube	0,2027	0,1930	0,0409	0,0597
controller	0,2288	0,1843	0,0551	0,0734
hammer	0,2857	0,2596	0,0414	0,0908
hand_cream	0,2155	0,2069	0,0256	0,0614
hard_disk_box_h	0,2512	0,2415	0,0624	0,1089
hard_disk_box_h	0,2405	0,2293	0,0430	0,0718
marker	0,2165	0,2101	0,0213	0,0451
pink	0,1856	0,1764	0,0172	0,0294
raspberry_box_h	0,2125	0,1774	0,0781	0,0932
raspberry_box_v	0,2995	0,2800	0,0443	0,0967
remote	0,1874	0,1732	0,0184	0,0298
screw	0,2821	0,2319	0,0364	0,0883
spray	0,2167	0,1999	0,0331	0,0478
tape_h	0,2281	0,1911	0,0337	0,1113
tape_v	0,1913	0,1736	0,0386	0,0397
white_rectangle_h	0,2057	0,1969	0,0375	0,1047
white_rectangle_v	0,1953	0,1910	0,0526	0,0647
wood	0,2561	0,2258	0,0389	0,0696
Total:	**0,2243**	**0,2058**	**0,0431**	**0,0830**

Table A.16: Results obtained with the *Complete-EgoHOS*-**Contact-GraspNet** configuration of the mean distance between the best grasp to the hand and the mean distance between the best grasp and all the other grasps of the same set.

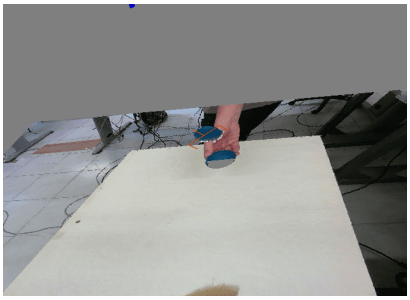
Appendix B

Grasp Generation Results

The following images display the best grasps generated for every object of the dataset. Results are obtained choosing the best grasp in terms of score after applying the pruning criteria defined in 3.3. The models used to retrieve these results are GraspNet [2] and Contact-GraspNet [4].

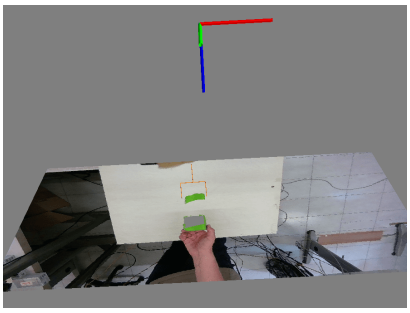
From the results, it is possible to notice some of the issues described in Chapter 4, for example, Figures B.2e and B.3g show a grasp interpenetrating the object point cloud, probably due to its size that is comparable with the gripper's maximum opening width, Figures B.3c and B.3e show the difficulties of finding a good grasp for the tape, whose segmentation mask is not precise.

(a) Ball with GraspNet.

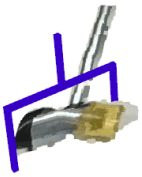


(b) Ball with Contact-GraspNet.

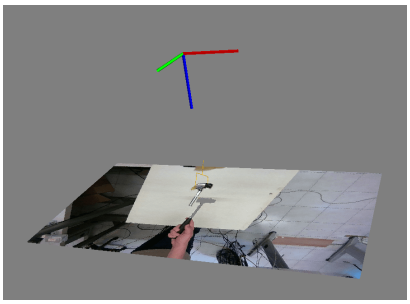
(c) Green_cube with GraspNet.



(d) Green_cube with Contact-GraspNet.

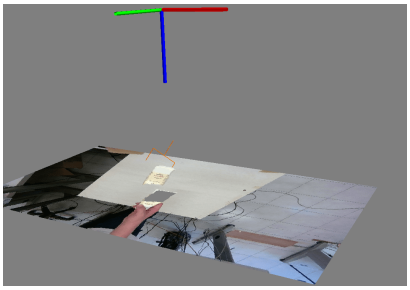


(e) Hammer with GraspNet.



(f) Hammer with Contact-GraspNet.

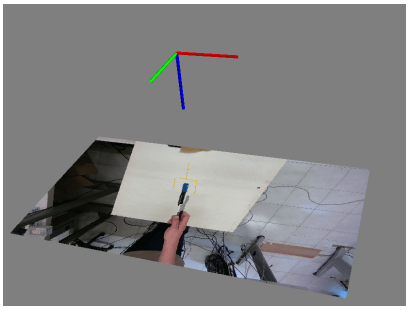
(g) Hand cream with GraspNet.



(h) Hand cream with Contact-GraspNet.

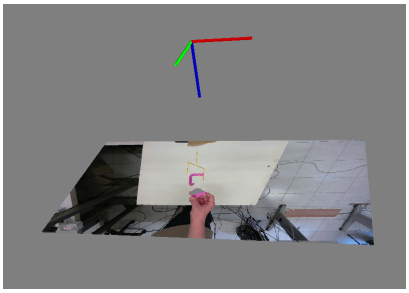
Figure B.1: Example of the generated best grasp on the objects belonging to the dataset. Other examples are shown in Appendix B

(a) Marker with GraspNet.



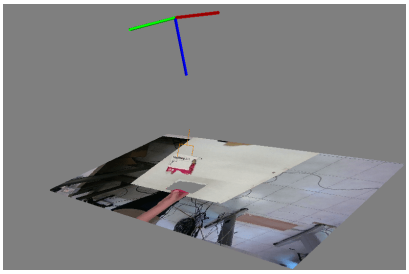
(b) Marker with Contact-GraspNet.

(c) Pink with GraspNet.



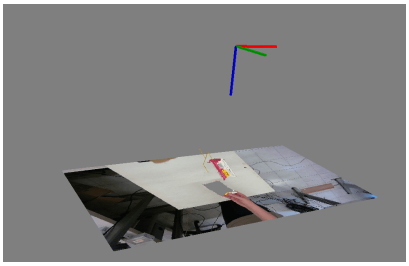
(d) Pink with Contact-GraspNet.

(e) Raspberry_h with GraspNet.



(f) Raspberry_h with Contact-GraspNet.

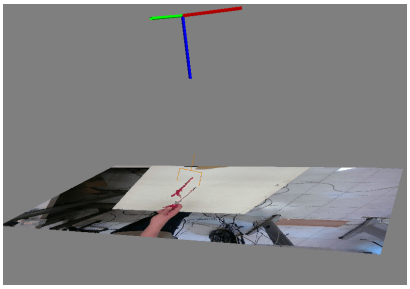
(g) Raspberry_v with GraspNet.



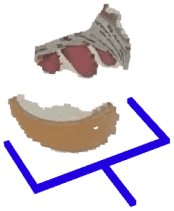
(h) Raspberry_v with Contact-GraspNet.

Figure B.2: Example of the generated best grasp on the objects belonging to the dataset.

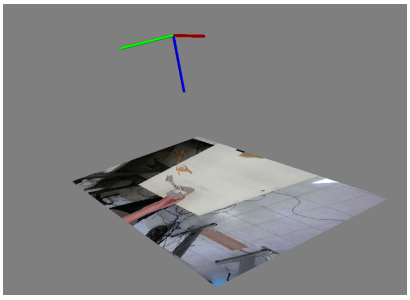
(a) Screwdriver with GraspNet.



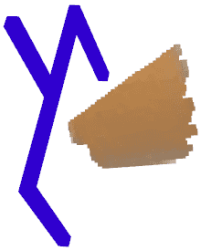
(b) Screwdriver with Contact-GraspNet.



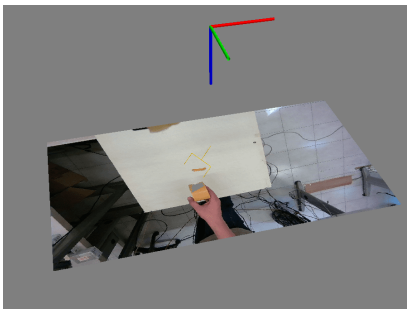
(c) tape_h with GraspNet.



(d) tape_h with Contact-GraspNet.

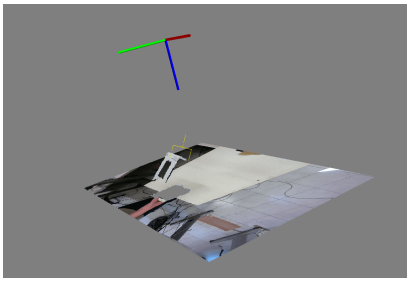


(e) tape_v with GraspNet.



(f) tape_v with Contact-GraspNet.

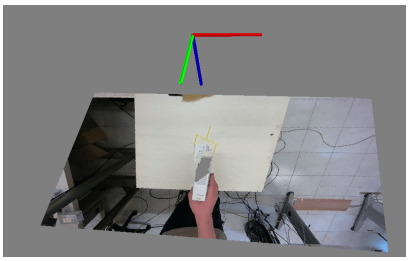
(g) hard_disk_box_h with GraspNet.



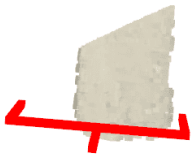
(h) hard_disk_box_h with Contact-GraspNet.

Figure B.3: Example of the generated best grasp on the objects belonging to the dataset.

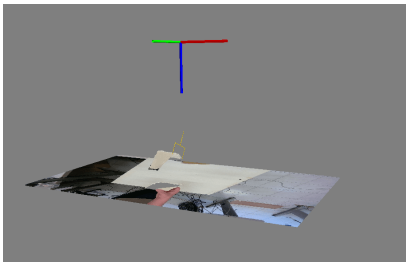
(a) hard_disk_box_v with GraspNet.



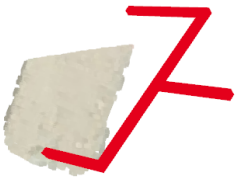
(b) hard_disk_box_v with Contact-GraspNet.



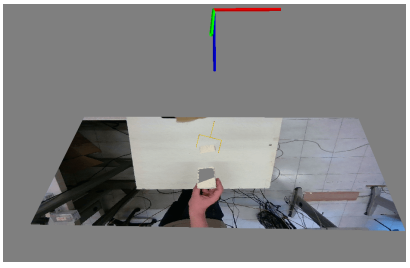
(c) white_box_h with GraspNet.



(d) white_box_h with Contact-GraspNet.



(e) white_box_v with GraspNet.



(f) white_box_v with Contact-GraspNet.

Figure B.4: Example of the generated best grasp on the objects belonging to the dataset.

Bibliography

[1] L. Zhang, S. Zhou, S. Stent, and J. Shi, *Fine-grained egocentric hand-object segmentation: Dataset, model, and applications*, 2022. arXiv: 2208.03826 [cs.CV].

[2] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-1billion: A large-scale benchmark for general object grasping," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 444–11 453.

[3] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *International Conference on Computer Vision (ICCV)*, 2019.

[4] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," 2021.

[5] M. D. Silva, R. Regnier, M. Makarov, G. Avrin, and D. Dumur, "Evaluation of intelligent collaborative robots: A review," in *2023 IEEE/SICE International Symposium on System Integration (SII)*, 2023, pp. 1–7. DOI: 10.1109/SII55687.2023.10039365.

[6] R. Sultanov, S. Sulaiman, H. Li, R. Meshcheryakov, and E. Magid, "A review on collaborative robots in industrial and service sectors," Nov. 2022, pp. 1–7. DOI: 10.1109/SIBCON56144.2022.10003014.

[7] S. Patil, V. Vasu, and K. Srinadh, "Advances and perspectives in collaborative robotics: A review of key technologies and emerging trends," *Discover Mechanical Engineering*, vol. 2, Aug. 2023. DOI: 10.1007/s44245-023-00021-8.

[8] M. Tavakoli, J. Carriere, and A. Torabi, "Robotics, smart wearable technologies, and autonomous intelligent systems for healthcare during the covid-19 pandemic: An analysis of the state of the art and future vision," *Advanced Intelligent Systems*, vol. 2, no. 7, p. 2 000 071, 2020. DOI: https://doi.org/10.1002/aisy.202000071. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aisy.202000071. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aisy.202000071.

[9] M. O. Yerebakan and B. Hu, "Human–robot collaboration in modern agriculture: A review of the current research landscape," *Advanced Intelligent Systems*, vol. n/a, no. n/a, p. 2 300 823, DOI: https://doi.org/10.1002/aisy.202300823. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aisy.202300823. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aisy.202300823.

[10] A. Kolbeinsson, E. Lagerstedt, and J. Lindblom, "Foundation for a classification of collaboration levels for human-robot cooperation in manufacturing," *Production & Manufacturing Research*, vol. 7, no. 1, pp. 448–471, 2019.

[11] S. Caldera, A. Rassau, and D. Chai, "Review of deep learning methods in robotic grasp detection," *Multimodal Technologies and Interaction*, vol. 2, no. 3, 2018, ISSN: 2414-4088. DOI: 10.3390/mti2030057. [Online]. Available: https://www.mdpi.com/2414-4088/2/3/57.

[12] M. Dong and J. Zhang, "A review of robotic grasp detection technology," *Robotica*, pp. 1–40, 2023.

[13] R. Newbury, M. Gu, L. Chumbley, *et al.*, *Deep learning approaches to grasp synthesis: A review*, 2023. arXiv: 2207.02556 [cs.RO].

[14] H. Duan, Y. Yang, D. Li, and P. Wang, "Human–robot object handover: Recent progress and future direction," *Biomimetic Intelligence and Robotics*, vol. 4, no. 1, p. 100 145, 2024, ISSN: 2667-3797. DOI: https://doi.org/10.1016/j.birob.2024.100145. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667379724000032.

[15] V. Ortenzi, A. Cosgun, T. Pardi, W. P. Chan, E. Croft, and D. Kulić, "Object handovers: A review for robotics," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1855–1873, 2021.

[16] B. León, A. Morales, J. Sancho-Bru, B. León, A. Morales, and J. Sancho-Bru, "Robot grasping foundations," *From robot to human grasping simulation*, pp. 15–31, 2014.

[17] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control* (Advanced Textbooks in Control and Signal Processing). Springer London, 2010, ISBN: 9781846286414. [Online]. Available: https://books.google.it/books?id=jPCAFmE-logC.

[18] C. Vesper, S. Butterfill, G. Knoblich, and N. Sebanz, "A minimal architecture for joint action," *Neural Networks*, vol. 23, no. 8-9, pp. 998–1003, 2010.

[19] W. Yang, C. Paxton, M. Cakmak, and D. Fox, *Human grasp classification for reactive human-to-robot handovers*, 2020. arXiv: 2003.06000 [cs.RO].

[20] N. Vahrenkamp, H. Arnst, M. Wächter, *et al.*, "Workspace analysis for planning human-robot interaction tasks," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 1298–1303. DOI: 10.1109/HUMANOIDS.2016.7803437.

[21] M. Pan, V. Skjervøy, W. Chan, M. Inaba, and E. Croft, "Automated detection of handovers using kinematic features," *The International Journal of Robotics Research*, vol. 36, p. 027 836 491 769 286, Feb. 2017. DOI: 10.1177/0278364917692865.

[22] C. Hansen, P. Arambel, K. Ben Mansour, V. Perdereau, and F. Marin, "Human–human handover tasks and how distance and object mass matter," *Perceptual and motor skills*, vol. 124, no. 1, pp. 182–199, 2017.

[23] H.-S. Fang, C. Wang, H. Fang, *et al.*, "Anygrasp: Robust and efficient grasp perception in spatial and temporal domains," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3929–3945, 2023. DOI: 10.1109/TRO.2023.3281153.

[24] P. Basili, M. Huber, T. Brandt, S. Hirche, and S. Glasauer, "Investigating human-human approach and hand-over," *Human centered robot systems: Cognition, interaction, technology*, pp. 151–160, 2009.

[25] E. Martinson, A. H. Quispe, and K. Oguchi, "Towards understanding user preferences in robot-human handovers: How do we decide?" In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017, pp. 516–521. DOI: 10.1109/ROMAN.2017.8172351.

[26] P. Rosenberger, A. Cosgun, R. Newbury, *et al.*, *Object-independent human-to-robot handovers using real time robotic vision*, 2020. arXiv: 2006.01797 [cs.RO].

[27] W. Yang, C. Paxton, A. Mousavian, Y.-W. Chao, M. Cakmak, and D. Fox, *Reactive human-to-robot handovers of arbitrary objects*, 2021. arXiv: 2011.08961 [cs.RO].

[28] C. Liu, W. Wang, R. Pang, C. Li, and Y. Shang, "6-dof grasp planning on point cloud for human-to-robot handover task," in *Cognitive Systems and Information Processing*, F. Sun, Q. Meng, Z. Fu, and B. Fang, Eds., Singapore: Springer Nature Singapore, 2024, pp. 73–86, ISBN: 978-981-99-8018-5.

[29] M. Gou, H. Fang, Z. Zhu, S. Xu, C. Wang, and C. Lu, "Rgb matters: Learning 7-dof grasp poses on monocular rgbd images," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13 459–13 466, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:232104937.

[30] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from rgbd images: Learning using a new rectangle representation," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3304–3311. DOI: 10.1109/ICRA.2011.5980145.

[31] H. W. Liu and C. Q. Cao, "Grasp pose detection based on point cloud shape simplification," *IOP Conference Series: Materials Science and Engineering*, vol. 717, no. 1, p. 012 007, 2020. DOI: 10.1088/1757-899X/717/1/012007. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/717/1/012007.

[32] A. ten Pas, M. Gualtieri, K. Saenko, and R. P. Jr., "Grasp pose detection in point clouds," *CoRR*, vol. abs/1706.09911, 2017. arXiv: 1706.09911. [Online]. Available: http://arxiv.org/abs/1706.09911.

[33] H. Liang, X. Ma, S. Li, *et al.*, "Pointnetgpd: Detecting grasp configurations from point sets," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, May 2019. DOI: 10.1109/icra.2019.8794435. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2019.8794435.

[34] Y. LeCun, B. Boser, J. Denker, *et al.*, "Handwritten digit recognition with a back-propagation network," *Advances in neural information processing systems*, vol. 2, 1989.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[36] K. O'Shea and R. Nash, *An introduction to convolutional neural networks*, 2015. arXiv: 1511.08458 [cs.NE]. [Online]. Available: https://arxiv.org/abs/1511.08458.

[37] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1512.03385.

[38] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[39] H. Zhang, X. Zhou, X. Lan, J. Li, Z. Tian, and N. Zheng, *A real-time robotic grasp approach with oriented anchor box*, 2018. arXiv: 1809.03873 [cs.RO]. [Online]. Available: https://arxiv.org/abs/1809.03873.

[40] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 922–928. DOI: 10.1109/IROS.2015.7353481.

[41] G. Riegler, A. O. Ulusoy, and A. Geiger, *Octnet: Learning deep 3d representations at high resolutions*, 2017. arXiv: 1611.05009 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1611.05009.

[42] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, *Pointnet: Deep learning on point sets for 3d classification and segmentation*, 2017. arXiv: 1612.00593 [cs.CV].

[43] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, *Pointnet++: Deep hierarchical feature learning on point sets in a metric space*, 2017. arXiv: 1706.02413 [cs.CV].

[44] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A survey on learning-based robotic grasping," *Current Robotics Reports*, vol. 1, pp. 239–249, 2020.

[45] A. Zeng, S. Song, K.-T. Yu, *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," *The International Journal of Robotics Research*, vol. 41, no. 7, pp. 690–705, 2022. DOI: 10.1177/0278364919868017. [Online]. Available: https://doi.org/10.1177/0278364919868017.

[46] D. Morrison, P. Corke, and J. Leitner, *Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach*, 2018. arXiv: 1804.05172 [cs.RO].

[47] J. Redmon and A. Angelova, *Real-time grasp detection using convolutional neural networks*, 2015. arXiv: 1412.3128 [cs.RO].

[48] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015. DOI: 10.1177/0278364914549607. eprint: https://doi.org/10.1177/0278364914549607. [Online]. Available: https://doi.org/10.1177/0278364914549607.

[49] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2022. arXiv: 1312.6114 [stat.ML].

[50] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[51] A. X. Chang, T. Funkhouser, L. Guibas, *et al.*, *Shapenet: An information-rich 3d model repository*, 2015. arXiv: 1512.03012 [cs.GR]. [Online]. Available: https://arxiv.org/abs/1512.03012.

[52] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics amp; Automation Magazine*, vol. 22, no. 3, pp. 36–52, Sep. 2015, ISSN: 1070-9932. DOI: 10.1109/mra.2015.2448951. [Online]. Available: http://dx.doi.org/10.1109/MRA.2015.2448951.

[53] J. Mahler, J. Liang, S. Niyaz, *et al.*, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," 2017.

[54] C. Eppner, A. Mousavian, and D. Fox, "ACRONYM: A large-scale grasp dataset based on simulation," in *2021 IEEE Int. Conf. on Robotics and Automation, ICRA*, 2020.

[55] D. Coleman, I. Sucan, S. Chitta, and N. Correll, *Reducing the barrier to entry of complex robotic software: A moveit! case study*, 2014. arXiv: 1404.3785 [cs.RO]. [Online]. Available: https://arxiv.org/abs/1404.3785.

[56] A. Kirillov, E. Mintun, N. Ravi, *et al.*, "Segment anything," *arXiv:2304.02643*, 2023.