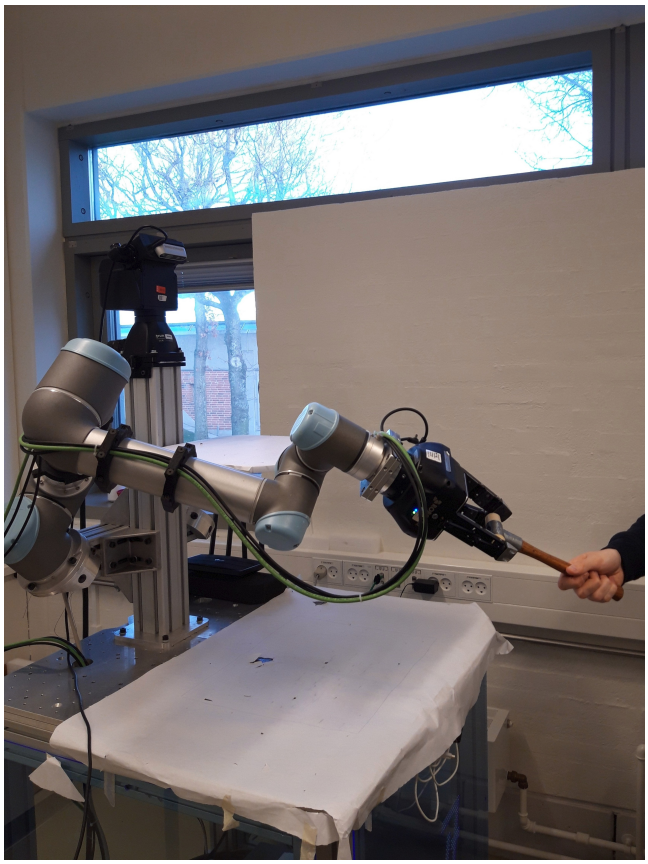
Task-oriented handover using task-agnostic grasping and affordance segmentation



9th semester project

Group 961

Aalborg University
Electronics and IT

AALBORG UNIVERSITY
STUDENT REPORT

Robotics
Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg
http://www.robotics.aau.dk

Title:
Task-oriented handover using task-agnostic grasping and affordance segmentation

Theme:
Robot-to-human handover

Project Period:
Fall Semester 2021

Project Group:
Group 961

Participant(s):
Daniel Lehotský
Marius W. Jørgensen
Albert D. Christensen

Supervisor(s):
Dimitris Chrysostomou

Copies: 1

Page Numbers: 77

Date of Completion:
February 25, 2022

Abstract:

This project develops a pipeline for task-oriented robot-to-human handover and implements it on a UR5 cobot equipped with an Intel RealSense D435 and a Robotiq 3-finger gripper. Task-oriented handovers are achieved by combining affordance segmentation and task-agnostic grasping. The affordance segmentation is performed by AffordanceNet and grasps are generated by GraspNet. Grasps and affordances are associated in a novel way, that allows for targeting a specific affordance of an object. This is combined with speech-to-text, which allows the user to request specific object to be picked up. The system performs task-oriented handover with a 72.22 % success rate on a custom object set.

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

Preface

This report was written by students of the 9th semester on the Robotics master. The project ran from the 1st of September 2021 to the 21st of December 2021. The project was supervised by associate professor Dimitris Chrysostomou from the department of Materials and Production, Aalborg University. This project aims to develop a pipeline for task-oriented robot-to-human handover that takes object affordances into account when grasping an object. This was done by creating a handover pipeline that combines affordance segmentation with task-agnostic grasp synthesis.

Group 961, Aalborg University

Marius W. Jørgensen
<mwja17@student.aau.dk>

Daniel Lehotský
<dlehot17@student.aau.dk>

Albert D. Christensen
<alchri17@student.aau.dk>

(1)

Contents

1 - Introduction

A robot-human handover is a fundamental skill for any robotic assistant. As such, it has recently been given a lot of attention [1, 2, 3]. Handovers are joint actions carried out between two agents, a giver and a receiver, where the goal is to transfer the object from the the giver to the receiver [4]. Robot-to-human handover is a subcategory of the robot-human handover, where the giver is the robot and the receiver is a human. The task of the robot is to stably present the object in an appropriate manner, and the objective of the receiver is to acquire the object by grasping and stabilising the object [5].

Humans excel at handovers, but in the field of robotics it remains an unsolved problem. To successfully carry out joint actions, the ability to share representations, predict actions, and integrate the predicted effects is needed [6]. Therefore, to design robots capable of intuitive handovers, many aspects have to be considered.

The first thing to consider is handover initiation. This initiation can either be handover by object request, where the receiver initiates the handover with a request, or handover by task request, where the giver is the one to initiate [4]. An example of the handover by task request can be a chef asking the sous chef to stir some food by offering the sous chef a spatula. Initiation is followed by the pre-handover phase which is then followed by the physical handover phase, see Figure 1.1.

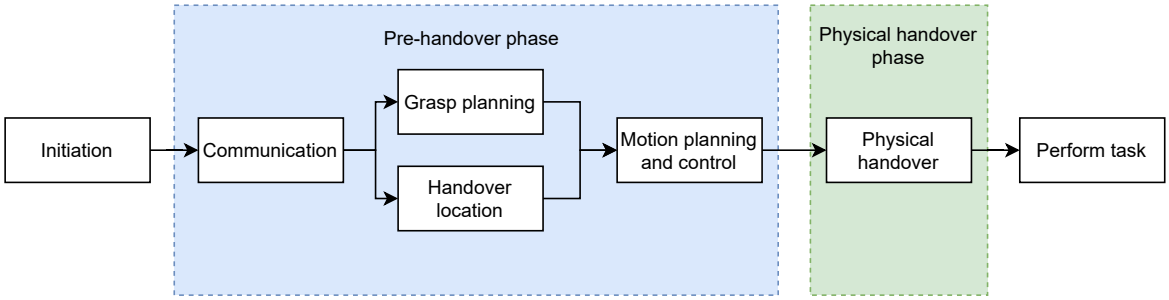


Figure 1.1: A generalized block diagram of a handover process.

In the pre-handover phase, the giver should anticipate the subsequent task the receiver will perform after the handover, which in turn affects how the receiver would like the object to be presented. These considerations inevitably affects the grasp planning. Once

the grasp is computed, motion planning is performed. From there the robot moves to the handover location. The proper handover location is dependent on both the object and the task. During handovers constant communication between the receiver and the giver causes the participants to continuously adjust their motion plans and the handover location. This communication can be indirect such as gaze and body language, or it can direct in the form of speech [4].

Once the receiver makes contact with the handover object, the pre-handover phase ends and the physical handover phase begins. This phase is mostly concerned with the stable exchange of the object. At this stage, it is crucial to continuously adjust the force applied to the object. As the receiver is increasing the force applied to the object, the giver should decrease the force. If timed correctly, the smooth handover is achieved. Otherwise, the object is released sooner than the receiver has a full control of it, resulting in the object falling to the ground [4].

As already mentioned, one of the key consideration of the giver agent is the subsequent task that is to be performed with the object by the receiver after handover. Ortenzi et al. suggest that when presenting an object for handover in an appropriate configuration that considers the subsequent task, the giver is more likely to be seen as a collaborative partner, which in turn improves the perception of the handover [7]. This is referred to as task-oriented handover. Depending on the subsequent task, the receiver might prefer the object to be presented with a certain orientation and specific parts free to grasp. For example, a knife would be grasped by the giver on the blade and presented with the handle free for the receiver to grasp.

Humans make use of a semantic understanding of object's functional properties in order to grasp the object appropriately for a given task [8]. The theory of affordances, as proposed by J. J. Gibson in 1977, lends itself very well to this problem. Gibson theorises that different objects allow different agents different functionalities [9]. Therefore, a human can perform different actions with an object than a robot depending on the robot's configuration.

The theory of affordances has gained traction in the field of robotics. Zhu et al. applied affordance theory to understand which tool to use to accomplish a given task [10], see Figure 1.2a. Koppula and Saxena applied affordance theory to anticipate human actions and predict their trajectories [11], see Figure 1.2b. Do et al. [12] used the segmented affordances of objects to perform robotic grasping, see Figure 1.2c.

This report investigated how affordances could be applied to the task of robot-to-human task-oriented handovers. For this report a task-oriented handover pipeline was developed by combining current grasping methods with affordance analysis. The system was developed with modularity in mind to allow for switching out modules. The pipeline allows for creating rules about how to grasp objects for handover based on affordances. One advantage of doing affordance-based rules is considering object function when handing over. Additionally, it allows for creating a few rules about a low number of

(a) (b) (c)

Figure 1.2: Previous work in the field of affordance analysis. **(a)** A human demonstrates to a robotic system the proper tool to crack a nut [10]. **(b)** A prediction heatmap of which affordances a person will utilize next [11]. **(c)** An object segmented into its affordances. Blue is a *grasp* affordance and red is a *contain* affordance [12].

affordances that apply to a much larger set of objects. The novel way of associating grasps and affordances developed allows for these rules to be made.

The contributions of this report to the field of task-oriented handover are the following:

- A novel full pipeline for a task-oriented handover.
- A novel method for associating task-agnostic grasps with affordances.

The system is implemented on a robot setup consisting of a UR5 cobot, a RealSense D435 camera, and a Robotiq 3-finger gripper as the end-effector tool. The robot setup is based on the Little Helper first proposed by [13], which has since seen modifications and is now the version 7 system located at Aalborg university [14].

2 - Problem Analysis

This chapter covers the analysis of the human-robot handover. It includes an analysis of the current methods used for achieving a smooth interaction between a human and a robot.

2.1 Related work analysis of the robot-human handover

This section will analyse the robot-human handover. As presented previously, a handover can be split into several stages where each has its distinct purpose to ensure a successful handover [4]:

1. Initiation
2. Preparation
3. Physical handover
4. Performance

A human-robot handover consist of these four steps. Each step can be solved with different methodologies. A lot of current work is being done in performing some or all steps involved in human-robot handover.

Koene et al. investigate if humans prefer speed or accuracy in the interaction of human-robot handover. It is found that humans are willing to trade off accuracy of the handover if it results in a faster human-robot handover. This is consistent with what is known from human-human interaction [15]. A novel approach for controlling handovers by tuning the timings of each step of the handover is proposed by Kshirsagar et al. They propose using signal temporal logic to ensure timing of each step and allow for tuning of timing parameters to adjust the handover action [16]. How humans handover objects and what humans consider to be a fluent interaction during handover are still research topics of the robot-human handover. Controzzi et al. find that human givers release their grip on objects at the same time as collision with the receiver happens. The speed with which the giver releases the object is found to be correlated with how fast the receiver is moving.

Additionally, a shorter release and handover duration when receiving an item from a robot is perceived as a more fluid interaction [17].

The general pipeline for human-robot handover that is used in literature consists of a grasp synthesis method, often a deep neural network, followed by trajectory generation to the handover location. The handover location is either a fixed position or dependant on the human receivers position and derived with a vision system [4, 1, 7, 16].

An example of a full pipeline for robot handover is presented by P. Rosenberger et al., who developed a novel approach for safety during the handover stage by detecting both the entire human and their hands with two different networks [1].

A human-robot handover is most commonly evaluated by the success rate of the handover action. Some human-robot handover implementations will also add a set of tasks to be completed after the handover. This is used to evaluate if the object is handed over in a way that allows for fast task execution. As a final step of testing, subjective measures such as questionnaires are used for evaluation. In rare cases, physiological measures of the test subjects are used for evaluation [4].

In a review of the field from 2020, it is identified that there is often a lack of consideration of the subsequent task to be performed with the object when robot-human handover is performed. Only looking at if the robot can pick up an object for handover is not sufficient to call it a success. Thus, task-oriented handover is an area where further work is needed [4].

2.1.1 Task-oriented handover

A task-oriented handover considers the subsequent task of the receiver once the handover is completed. An example of this could be a tool handover. If the tool is presented to the receiver such that the receiver can grasp the tool's handle right away, the receiver can continue with their task without disturbance. Ortenzi et al. [7] found that the perceived quality of the handover and subsequent task performance increased with task-oriented handovers as compared to a task-agnostic handover. According to Brown and Sammut [18], four aspects must be considered for robotic task-oriented tool usage. These are:

1. Understanding the effect that the tool usage achieves.
2. Identifying the properties of an object that makes the tool suited for the given task.
3. Determining the correct position and orientation of the tool prior to usage.
4. How the tool should be manipulated.

A traditional handover has to consider trajectory planning, grasping, handover location, and handover orientation [4]. Task-oriented handovers differ by doing task-oriented grasping and being able to consider handover orientation based on the object function,

while still considering trajectory planning and handover location. The majority of the currently available research has investigated each sub-problem separately, and the following sections will provide a short overview of research performed in the field so far.

Trajectory planning

A handover motion planning can be split into two segments - trajectory and grasp planning. Trajectory planning is the process of moving the robotic arm from an initial configuration to the grasp point. Trajectory planning is followed by grasp planning, which generates an object-picking action. Due to the closeness between the robot and the operator, trajectory planning is an important aspect of the handover to consider. Ortenzi et al. suggest that a robot's trajectory during a robot-human joint action such as robot-to-human handover should ideally follow trajectories that are legible, predictable, safe, but also robust, reactive and context aware [4]. Moreover, a study by Cakmak et al. finds that humans prefer human-like robot configurations as these configurations are more readable [19]. In a study by Huber et al. [20], test participants showed lower reaction time and increased feeling of subjective safety when cooperating with the robot following a minimum jerk velocity profile of the end-effector. It was also observed that when using a minimum jerk profile, the robot could move at a higher speed without making the receiver uncomfortable. The authors of the study argue that the minimum jerk velocity profile can be considered more biologically similar to humans than the traditional trapezoidal velocity profile in joint space. Therefore, a human-like motion is a desired aspect of the handover.

Rasch et al. have published three papers on the topic of the human-like handover trajectory. In their first paper [21], a Cartesian space trajectory model based on IMU data collected during a handover task between two people was proposed. According to their experimental results, a proposed motion model fits the natural human trajectory better than other motion models found in the literature. In their second work [22], a joint motion model based on the motion profiles of the human joints, specifically elbow flexion/extension and shoulder flexion/extension, was developed. The main goal of this paper was to develop a joint motion model that can produce trajectories that are recognisably human. The model was tested with both a humanoid Pepper robot and a industrial Kuka Youbot. The results of the experiment showed that the proposed joint motion model increases a feeling of safety when used for a handover with the Kuka robot. The same effect was not recorded for the Pepper robot. In their third paper [23], the authors proposed a combined model which allows for human-like joint configurations while planning in the Cartesian space.

Handover location

In the study by Parastegari et al. [24], a handover location is studied based on the human-to-human handover demonstrations. Based on the data collected during the human study, they propose a dyadic joint torque model. In the human-robot experiment, it is confirmed

that the proposed model can generate the handover positions that match the preferred height and acceptable distance from the human participant. The model takes into account the placement and kinematics of the human along with object weight.

In the aforementioned study, users showed preferences over varying handover locations, with the height of the handover location being more important than the distance from the user. This implies that a good handover location can improve the user's experience during the handover.

Handover orientation

The handover orientation describes how the giver orients the object at the handover location when presenting it to the receiver. Finding a good orientation is challenging because it depends on the properties of the object and the state of the receiver. Early work into this problem by Aleotti et al. [25] proposed an analytical method for calculating the object orientation such that an object's handle, segmented from a point cloud, would always be pointed towards the reciever. Their results suggest that by taking into account the affordance of an object, the perceived quality of the handover is increased. A limitation of the proposed method is that each object has to be manually annotated with a desired handover orientation.

In a study by Chan et al. [26], human handover demonstrations are used to learn proper handover orientations. In this study, the authors grouped the collected handover orientations with their respective affordances, proving that orientations are related to the affordances of the object. Another study done by Chan et al. [27] investigates the handover orientations as demonstrated by humans. Based on the collected data, an optimization method for calculating mean handover orientations was proposed. Furthermore, the paper introduces a concept of affordance axes which can be used to compare handover orientations. In their more recent paper, Chan et al. [28] propose a novel method for computing the mean handover orientation based on data collected from human handover demonstrations. Rather than labelling the data manually, a model learns the object's affordance axis from the data.

Previous work in handover orientation suggests that the proper orientation of an object is related to the affordances or functions of an object. Affordance theory can therefore improve the robot-to-human handover in relation to the orientation of the object.

Task-oriented grasping

Several methods have been proposed for successfully grasping objects [29, 30, 31]. The aim of these methods is to stably grasp an object, therefore, the generated grasps should optimize for grasp stability. Such methods does not consider the task when generating grasps, and as such they are task-agnostic. In order to perform task-oriented handover, task-oriented grasping is needed. Task-oriented grasping refers to a grasp choice based not only on the stability of the grasp itself but also the semantic understanding of the

object and the subsequent task. To illustrate an example for a handover task, a screwdriver should be grasped by its metal rod, because the operator prefers to grasp the screwdriver by its handle. A multitude of methods for this problem were proposed in recent years. The remainder of this section will investigate these methods.

Task-oriented grasping methods can be categorized into wrench based methods and data driven methods. Wrench based methods both generates grasps and models the tasks in wrench space, and often propose a task-oriented grasp metric [32, 33, 34]. Data driven methods have largely replaced other methods such as wrench based. Early data driven methods often made use of small datasets that scaled poorly and were limited to a few select objects [35, 36].

In the recent years, deep learning methods have been the main approach to task-oriented grasping. Methods using affordance theory are by far the most popular. The methods either learns affordances explicitely or implicitely.

Grasping with explicit affordance knowledge

Explicit affordance grasping methods often approaches the problem by solving it as a two-stage problem. In the first stage, the object affordances are either segmented on a pixel level or predicted as discrete attributes, as shown in Figure 2.1. Task-oriented grasps that grasps the different object affordances are then generated based on this explicit knowledge.

(a) **(b)** **(c)**

Figure 2.1: (a) With convolutional network capable of segmenting a scene into its affordances, the transport affordance has been segmented in green [37]. **(b)** A hammer segmented into the grasp and pound affordance with AffordanceNet [12]. **(c)** A saw segmented into its *grasp* and *cutting* affordances. [38]

Nguyen et al. [39] performed task-oriented grasping using a two stage-method. The first stage performs affordance segmentation on detected objects in an RGB image. The pixel level affordance masks are then post-processed with a dense conditional random field. In the second stage, rectangle grasps are computed using a rule based approach, where the mean pixel of an affordance is the grasping position on the detected object affordances. Real world robotic applications showed that they could grasp certain affordances such as *grasp* and *contain* with a 90 % success rate with a WALK-MAN robot.

Detry et al. [37] also perform task-oriented grasping in two stages. They trained a MultiCNN neural network on a small synthetic dataset to segment affordances in depth images, and thereby get the object part that could be used to carry out a task. In the second

stage, they fit pre-defined grasping shapes to the image in order to find suitable grasps for the viable parts of the image. The segmentation neural network was trained on a synthetic dataset of 3D object meshes. Individual parts of a mesh were annotated with a task that grasping that specific part of the object affords. The grasping neural network was trained using kinesthetic teaching, which is time consuming. The solution was implemented on a 7 DOF robot equipped with a Kinect 1 camera and a three-fingered gripper. With their method, the authors were able to successfully grasp different affordances, given a task and an object placed at random, with a success rate of 69 %.

Kokic et al. [40] also implemented task-oriented handover in a two-stage method using affordance theory. The first stage does affordance segmentation, object detection, and orientation estimation on point cloud data using an AFF-CNN and a CO-CNN network respectively. In the second stage, the appropriate grasps are generated based on the object detection and orientation, which is used to calculate the approach direction. This method requires knowledge of both the object and orientation which limits its applications in the real world. Furthermore, the method was only tested in a simulation.

Ardón et al. [41] propose a new dataset that combines grasping and affordance, called grasp affordance. Using Markov logic network trained on the proposed dataset, the authors can predict grasps that directly relates to the success of a task.

Chu et al. [42] recognized that previous affordance explicit based task-oriented grasping methods was limited by the amount of object classes available in affordance datasets. Therefore, they implemented a deep neural network for category-agnostic affordance segmentation. They do not detect object classes, instead they detect foreground and background, which means that the proposed solution also generalises to unseen object categories. Moreover, the algorithm allows for multiple ranked affordance predictions for the same object part. For example, a bowl has "contain" as its primary affordance. However, it is also possible to use the bowl as a scooping tool, which means "scoop" is bowl's non-primary affordance. This was later demonstrated in a physical manipulation experiment, in which the robot was tasked with scooping some beans. Using a rule based grasp generation strategy, the robot was able to carry out this task using both a trovel, which has scoop as its primary affordance, and a bowl. The solution was capable of grasping the *grasp* affordance of 6 different objects with a success rate of 85 %. Later, Chu et al. [43] revisited the problem and proved it is possible to segment category-agnostic affordances by training on a synthetic dataset, and finally demonstrating its use with a real world robotics application.

Lastly, Liu et al. [44] recently introduced a new grasping network Context Aware Grasping Engine, CAGE in short, which is built on a task-oriented grasp discriminator using a Wide & Deep learning model. A dataset was generated by sampling grasps with an antipodal grasp sampler [45] and then evaluated using their method and then labelled with an affordance which was the nearest affordance measured in Euclidian space. The affordance were predicted using the AffordanceNet method [12]. CAGE's grasp selection is

based on the object's segmented affordances, predicted object's material, predicted object's state and a desired task.

Grasping with implicit affordance knowledge

Implicit affordance methods does not expressly segment an object into its various affordances, nor does it predict the attributes. Instead, this step is skipped and task-oriented grasps are either predicted directly or first sampled with a task-agnostic method and then evaluated with a task-aware quality function. These methods are affordance implicit since they implicitly learns the affordances of objects. Affordance implicit methods, therefore, does not provide any explicit information about an objects affordances which could be utilized in manipulation policies after having grasped the object.

Cavalli et al. [38] propose an evaluation quality metric for task-oriented grasping. This novel metric describes how good a grasp is for a given task using a novel affordance function, which is manually defined for each affordance. For a given grasp G on an object O and a point U located on the surface of O, the function $F_T : (O, G, U) \mapsto \mathbb{R}$ can be computed. Note, that the point U is located within the object's area of use, e.g. a hammer head for hammer, or a blade for a knife. The best grasp is the one that maximizes the function F_T. The metric is calculated through a vision system. This method scales poorly since each affordance function has to be manually formulated, furthermore this method is limited to a simulation for now.

Din et al. [46] also developed a quality metric for task-oriented grasps. For a dataset of 20 objects, grasps were generated using a task-agnostic grasp generator. Each grasp where then manually annotated as valid or invalid for a specific task, i.e. affordance. The grasp is then encoded within a feature vector, which contains information about the grasp's position, grasp's orientation, object (e.g. cup), task (e.g. grasp-to-pour), and a binary class label (i.e. valid or invalid). From this dataset a set of binary discriminators were trained, that could label a grasp as either valid or invalid for a given task.

Murali et al. [47] created a dataset called TaskGrasp consisting of 250,000 task-oriented grasps for 56 tasks and 191 objects. The dataset was created by sampling grasps with a task-agnostic grasping generator [45], and labelling each sampled grasp with a specific task by hand. From this dataset, the authors trained task-oriented grasping network similar to the task-agnostic grasp generator [30] to compute task-oriented grasps from a point cloud and a task in a single step.

Fang et al. [48] suggested to evaluate grasps by their task success rate. They extended the task-agnostic grasp generator GQ-CNN [29] by evaluating the success rate of each proposed grasp in a simulation. This method optimizes a grasp discriminator function and a manipulation selection function in parallel. The method suffers from scaling issues as manipulation functions for each new task has to be defined in order to teach the task to the system.

Kokic et al. [49] created a framework that can estimate the pose of hands and objects from the GUN-71 dataset, which contains video sequences of humans performing tasks

from an egocentric point of view. They then annotated each video sequence with a task label. With this framework, they created a dataset of task-oriented grasps in a voxelized 3D space. From this dataset a model named TOG-T was trained, which could generate task-oriented grasps on novel objects not seen in the dataset. They argue that this method simplifies the data collection and scales well.

Challenges and trends in task-oriented grasping
Most of the current methods utilizes affordance theory in task-oriented grasping. Majority of the methods are affordance explicit where an object's affordance is predicted as either attributes or a semantic map, this information can then be used for making manipulation policies such as handover orientation. The affordance explicit methods mostly rely on rule-based methods for grasping, which does not take into account the stability of the grasp [39, 43, 42], or methods that suffers from scaling issues [37, 40]. Affordance implicit methods skips the affordance prediction step and instead either generate task-oriented grasps in a single step, or samples task-agnostic grasps and then discriminate between the sampled grasps with a trained task-oriented grasp evaluator. Affordance explicit methods has the advantage of enabling affordance based manipulation policies. Using an affordance explicit method in combination with a task-agnostic grasping might provide more stable grasping and enable affordance based manipulation policies at the same time.

Currently it is a challenge to compare the methods against each other, as the system setup varies between the studies in terms of hardware, objects to be grasped, manipulation experiments and definitions of success rate, but success rates appears to be between 69 % and 95 % depending on the task and objects.

This covers some of the related work in the field of the task-oriented handover. The following sections will cover individual parts required for executing a robot-human handover, such as grasp synthesis, affordance prediction and handover evaluation. Ideally, motion planning should also be one of the investigated topics. However, although essential, motion planning is not the focus of this report, and therefore will not be investigated further. Instead, the ROS native MoveIt framework is selected as the motion planning method.

2.2 Task-agnostic grasp synthesis

Section 2.1.1 suggests that task-oriented handover can be solved with affordance explicit methods in combination with task-agnostic grasping. This section therefore investigates task-agnostic grasp synthesis.

2.2.1 Examining task-agnostic grasp synthesis

Grasp synthesis is the task of producing gripper configurations that are capable of grasping and holding objects [50]. The objective is to stably grasp an object. Task-agnostic grasp synthesis methods, therefore, aims to synthesize grasps that are as stable as possible. This section investigates what defines a robotic grasp and how the grasps are evaluated.

Defining A Robotic Grasp

A grasp representation describes where and how the robot's gripper should be placed, oriented and configured in order to perform a grasp. When calculating a grasp representation, the constrains of the gripper, such as its size and kinematic configuration, must be considered. Therefore, not all grasp representations might be applicable to all robotic grippers. Furthermore, when synthesizing a robotic grasp, an object's constraints, such as shape, size, and geometry, must also be considered [4]. The following robotic grasp representation investigation is, therefore, limited to grasp representations applicable to parallel grippers, see Figure 2.2, or grippers that can act as parallel grippers.These types of robotic grasps are sometimes referred to as antipodal grasps. Antipodal points are any two points on an object's surface with normals that are collinear but in the opposite direction. Grasping the object using the antipodal points on its surface guarantees force closure [51]. The force closure refers to the state when all of the object's motion is completely or partialy restrained due to the large enough constraint forces being applied to the object by the grasping device, i.e. the robot's gripper [52].

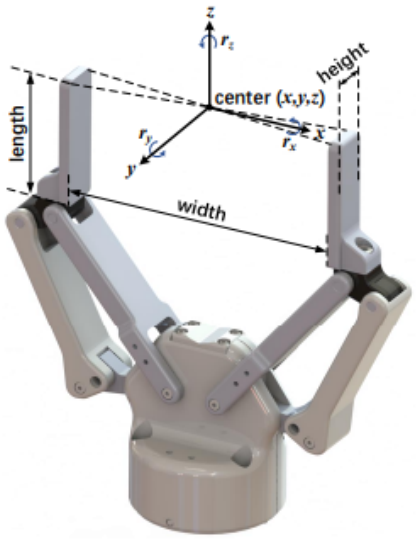
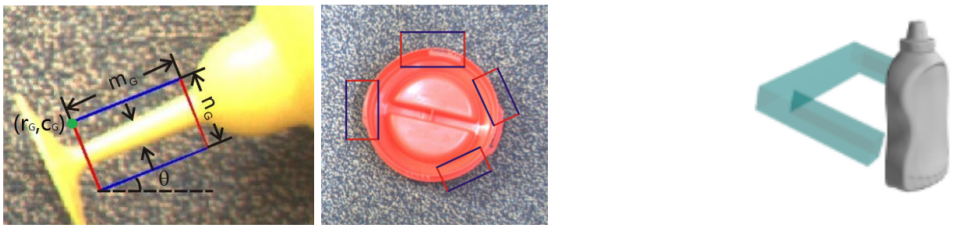


Figure 2.2: A parallel gripper with two diametrical fingers [53].

A widely used category of grasp representations is the 2D grasps or sometimes known as the grasp rectangle. There exist many 2D grasp representations. Common to them is that they are well suited for images where the synthesized grasp location can directly be computed from the 2D pixel position and are further augmented with an orientation. The authors of the Cornell dataset proposed a popular 2D grasp representation in 2011 [54], which besides specifying a position and orientation in the image plane also specifies the opening and width which has the shape of a rectangle, see Figure 2.3a. This seems to be the most widely used 2D grasp representation.

The 3D representation is the other widely used grasp representation. The basic 3D representation consists of a 3D position (x, y, z) and a 3D orientation. There exist no unified, or widely agreed upon, 3D grasp representation, as is the case for 2D representations. Instead, different methods apply their own grasp representations. Nevertheless, they are often visualized as a simple 3D model, see Figure 2.3b



(a) **(b)**

Figure 2.3: (a) A visualization of a grasping rectangle from the Cornell grasping dataset [54]. **(b)** A 3D parallel gripper grasp visualized [55].

Evaluating robotic grasping

There exist no unified approach for evaluating proposed robotic grasping methods at the time of writing this report. Nevertheless, there are two main approaches. One is to evaluate the grasp synthesis method on grasp synthesis datasets, and the other is to evaluate the ability to grasp physical objects.

Evaluating on grasp synthesis datasets

One method of evaluating and comparing the performance of a grasp synthesis method is to evaluate it on the dataset it was trained on. This is popular among the 2D grasp synthesis methods [56][57]. Since these methods are often trained on the Cornell Grasping Dataset, this dataset has become a benchmark dataset.

More recently, GraspNet-1Billion [58] proposes the first large scale dataset for 3D grasp synthesis, which, as the Cornell dataset, can also be used for evaluation [53].

While these approaches to evaluation have issues, such that they might not be indicative of real world performance, they evaluate on the same dataset which allows for easy comparison.

(a) (b)

Figure 2.4: Different grasp synthesis object sets. **(a)** The tool items subset of objects from the YCB dataset [61]. **(b)** All 42 objects from the ACRV dataset [62].

Evaluating On Physical Object Test Sets

The other approach to evaluating grasp synthesis methods is to test the performance on real life robotic systems. Again, there exist no unified framework for evaluating the performance of grasping methods on real robotic systems [59]. Most proposed methods are evaluated with the grasping success rate, though this metric might be poorly defined between various methods. The authors of this report adopts the success rate definition of [29] which is further modified for this project and is given as: "The percentage of grasps that were able to lift and hold a desired object during transportation".

The grasping success rate is difficult to compare between methods since each method tests on different object sets. Some use their own [60]. Although test objects are often chosen to reflect common "household" items, it remains a poorly defined category. The YCB [61] and ACRV [62] datasets, see Figure 2.4a and Figure 2.4b, are standardized object sets which solve the problem of non-standardized objects for testing. Though researchers cannot agree on which objects from the two object sets to use [58, 55, 59], with [58] and [59] using different subsets of the YCB or ACRV object sets in combination with their own objects.

Currently there is no unified method for evaluating grasping. But the authors of this report adopt the success rate metric. The household or tool subset of the YCB object set is well suited for a grasp method comparison.

2.2.2 Task-agnostic grasp synthesis methods

Grasping can be planned using different modalities[63]. The most common domain for grasp synthesis is the vision modality, where grasp poses are synthesised from 2D or 3D images.

Several methods have been proposed to solve the problem of grasp synthesis in the vision domain. These methods can be categorized into two broad categories - analytical

and data driven [64]. Analytical methods utilize traditional computer vision techniques that analyze the shape, size and geometry of objects in order to predict suitable grasp candidates[65, 66, 34]. However, with the increasing popularity of deep learning, data driven methods have largely replaced the analytical approaches [64, 67].

Model-based approaches

The data-driven methods can further be categorized into *model-based* and *model-free* approaches. Model-based approaches require prior understanding of objects in order to generate grasping poses. Model-based approaches are split into three steps - pose estimation, grasp synthesis, and planning of a feasible grasp path or trajectory [64]. By first detecting and estimating the pose of an object, pre-defined grasps are synthesised based on information of the objects such as CAD, URDF or model scans [68, 69, 70].

While model-based approaches perform well in challenging scenes such as clutter, they suffer from a lack of generalization, meaning that they do not perform well on novel unseen objects.

Model-free approaches

Model-free methods have been proposed to deal with the model-based lacking ability to generalize to new objects [71]. Model-free methods do not require any prior knowledge of objects in order to grasp them, and are, therefore, better suited for generalized autonomous grasping. Model-free approaches can be split into *supervised learning* and *reinforcement learning* [64].

Reinforcement learning

Reinforcement learning has proven capable of grasping novel unseen objects [64]. However, the reinforcement learning approaches suffer from the drawback that obtaining a sufficiently large dataset is a time consuming obstacle to the adaptation of reinforcement learning approaches [72].

Supervised learning

Supervised methods learn grasps from one or more datasets using machine and deep learning methods. Supervised methods can either be discriminative or generative, see Figure 2.5. Discriminative methods generate a set of grasp candidates which then gets evaluated using a quality function in order to chose the best candidate. This method can be computationally expensive. On the other hand, generative methods only predict one grasping pose, which is, thus, faster than discriminative methods [64].

Model-free methods were first proposed by [71], who recognized the need for object independent grasping. The proposed method generated two grasping points for a parallel gripper in 2D RGB images and then triangulated the 3D positions of the points. More recently, it has been proven possible to predict 4-DOF grasping rectangles to pick objects

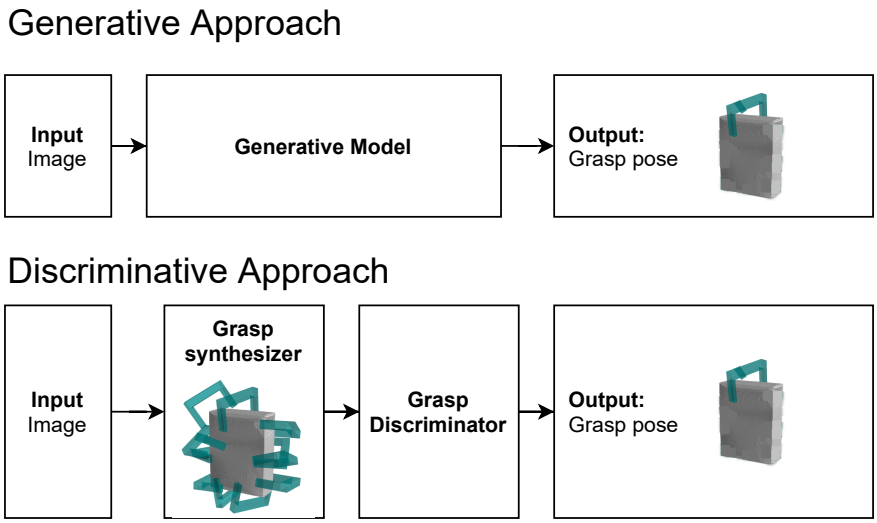


Figure 2.5: A visualization of the generalized difference between the generative and discriminative approach. Uses modified illustrations from [55]

from RGB and RGB-D information using deep learning [56, 57, 29, 59], with the GG-CNN method from [59] achieving state-of-the-art performance on the YCB household dataset in 2018 with a grasp success rate of 92 %. This has since been outperformed by [31] with a grasp success rate of 92 $\pm$ 5 % on static objects.

Other methods predict 6-DOF grasping poses in RGB and RGB-D images[60, 30, 53]. This is advantageous as it allows a robot to grasp objects from other perspectives than top-down. The top-down perspective is ill-suited for when the objects of interest are occluded by other objects, which might affect the success of the grasp [30]. A recent trend in grasp synthesis is to generate grasp poses from point clouds using networks such as PointNet [73] or PointNet++ [74], completely independent of prior object information. PointNetGPD recently achieved results of 89.33 % grasping success rate on a set of house hold objects using 6-DOF grasping poses, while also achieving good results on sparse point clouds which negates the influence of errors from the camera [55]. Fang et al. also proposes a discriminative grasping network, named GraspNet, trained on their own dataset, GraspNet-1Billion, which can perform dense 6-DOF pose prediction in challenging cluttered scenes by basing their network structure on PointNet++. According to the authors, this method performs well when grasping novel objects which it has not been trained on [58].

Discriminative methods seem the most suited for affordance explicit task-oriented grasping. Discriminative methods generate a selection of grasp with a quality score for each grasp. This is beneficial as each grasp can be associated with an affordance and then evaluated on the grasp score. Generative methods only produce a single grasp, which might not grasp the desired affordance, or the suggested grasp might not be feasible since the methods presented do not take the the reachable workspace into account. Furthermore,

6-DOF grasp synthesis methods are better at dealing with occlusion. For these reasons, methods such as GraspNet and PointNetGPD seems the most suited for this project.

A summary of the grasp synthesis methods discussed in this section is presented in Table 2.1.

Method	Year	Model-based	Model-free	Discriminative	Generative	4-DOF	6-DOF	YCB dataset	Unknown or non-standardised dataset
— [70]	2017	✓							
DOPE [69]	2018	✓							
PPR-Net [68]	2019	✓							
— [71]	2007		✓	✓			✓		87.7 %
GQ-CNN (Dex-Net 2.0) [29]	2017		✓	✓		✓			93 %
6-DOF GraspNet [30]	2019		✓	✓			✓		88 %
PointNetGPD [55]	2019		✓	✓			✓	82 %	
Grasp-it like a pro [60]	2020		✓	✓			✓		86.7 %
GraspNet [58]	2020		✓	✓			✓		
AVN-FAS [53]	2021		✓	✓			✓		91.1 %
Single-grasp [56]	2015		✓		✓	✓			
HCF [57]	2017		✓		✓	✓			
GG-CNN [59]	2018		✓		✓	✓		92 %	
GG-CNN2 [31]	2020		✓		✓	✓		92 %	

Table 2.1: A summary of the investigated grasp synthesis methods.

2.3 Affordance segmentation

Section 2.1.1 suggests that task-oriented handover can be solved with affordance explicit methods. This section therefore investigates affordance segmentation of objects.

2.3.1 Examining affordance segmentation

Affordance analysis, first proposed by Gibson [9], is a theory that different objects and places affords different actions depending on the agent. Gibson himself remarked, “If you know what can be done with a[n] object, what it can be used for, you can call it whatever you please”. Visual affordance is the task of extracting affordance information from either

a video or images. Affordance segmentation is part of visual affordance and has seen a lot of attention in the recent years.

Affordance segmentation is a sub-category to the affordance analysis inside the computer vision domain. Affordance segmentation is the task of predicting the affordance label of each pixel in an image, or a point in a point cloud [75], see Figure 2.6. Each pixel can either have a single or multiple labels, which represent the affordance that group of pixels allows the agent.

Figure 2.6: An example from the UMD dataset [76]. Each object are segmented into different functional parts, such as a handle for grasping and the saw blade for cutting.

In case of the affordance segmentation in objects such as tools and household objects, what affordances that is predictable depends on the datasets available. While several affordance segmentation datasets exist, see Table 2.2, the UMD dataset [76] and the IIT-AFF dataset [39] are the most widely used datasets for benchmarking affordance segmentation methods.

Affordance segmentation shares many similarities with other computer vision segmentation tasks. Affordance segmentation often uses its own evaluation metrics, not comparable to evaluation metrics of the other computer vision segmentation tasks. Amongst these, the weighted F_β^w-measure metric, see equation (2.1), as proposed by Margolin et al. [79], is the most widely used.

$$F_\beta^w = (1+\beta)\frac{Precision^w.Recall^w}{\beta^2.Precision^w + Recall^w} \tag{2.1}$$

The F_β^w-measure metric is an extension of the F_β-measure. Both measures share the β parameter, which commands the preference between complete detection versus over-detection. If $\beta > 1$, then recall has a higher weight than precision. Similarly, if $\beta < 1$, then recall weighs less than precision. Lastly, $\beta = 1$ means both recall and precision are weighted equally.

Dataset	Year	Size	Modality	Labels
UMD [76]	2015	10,000	RGB-D	*grasp, cut, scoop, contain, pound, support, wrap-grasp*
IIT-AFF [39]	2017	24,677	RGB-D	*contain, cut, display, engine, grasp, hit, pound, support wrap-grasp*
CAD-120-Affordance [77]	2017	9916 images	RGB	*open, cut, contain, pour, support, hold*
3D-AffectNet [78]	2021	23,000 shapes	Point cloud	*grasp, lift, contain, open, lay, sit, support wrap-grasp, pour dislay, push, pull listen, wear, press cut, stab, move*

Table 2.2: A list of popular affordance segmentation datasets.

F_β^w score expands on the F_β-measure by:

1. extending the definition of True Positives, True Negatives, False Positives, and False Negatives to non-binary values,
2. weighting the errors based on their location and neighbourhood.

Affordance segmentation shares a lot of the same problems and challenges as other computer vision segmentation tasks, since it is often solved with the same methods. These challenges or issues are typically illumination, occlusion, clutter and viewpoint variations. Furthermore, the affordance labelling of objects is challenging, since widely different looking part of objects might share the same label, such as handles of a bagpack and a handle of a screwdriver. In contrast, the same part can also have multiple labels [75].

2.3.2 Methods for affordance segmentation

Affordance segmentation is a subtask within computer vision, and as such previous work has made great use of computer vision techniques especially object detection and pixel-wise segmentation [75]. While no longer the primary method for performing affordance segmentation, traditional computer vision techniques in combination with traditional machine learning methods such as SVM and forest classifiers have been used to assign single-label affordances to each pixel. Using geometric features computed with tradtional computer vision techniques from RGB-D images, A. Myers et al. [76] achieved the state-of-the-art results on the UMD dataset [76] in 2015.

With the rise of deep learning, auto-encoders have been applied to the affordance segmentation problem [80], and variations of popular object detectors such as Fast-RCNN [81] and instance segmentation networks such as Mask-RCNN [82] have also proven successful in solving the task [39, 37, 12, 42]. R. Detry et al. [37] predicts single label affordance in depth images as they argue geometric information is more relevant for inferring affordance. They do so using a three-branch decoder and a common encoder CNN. In the same manner, Nguyen et al [39] also predicts a single-label affordance using a R-FCN network in RGB-D images. AffordanceNet [12] proposes a modified Mask-RCNN structure for detecting objects and affordance segmentation in parallel, with this method they achieve state-of-the-art results on the UMD dataset. A method for assigning multiple affordance labels to each pixel has been proposed by [42] in RGB images. This allows them to assign primary and secondary affordances to each object, which in turn allows them to solve complex tasks where they can chose suitable tools based on either their primary or secondary affordance. Chu et al. [83] revisited the problem and recognized that previous affordance recognition methods performed poorly on novel objects. They modified their method to only detect foreground and background, thereby making the method able to generalize to novel objects. They named this method AffordanceNet-Context, and they achieved state-of-the-art results on the UMD dataset for novel object categories with an F_β^w score of 0.69. More recently, Deng et al. [78] argues that the 3D domain lends itself better to affordance prediction as 3D only methods such as the PointNet++ [74] are better at exploiting geometric properties compared to 2D and 2.5D methods. To this end, a 3D only dataset based on the ShapeNet database was created and used to train their model based on PointNet++ [78]. This report considers AffordanceNet [12] as the state-of-the-art method for known object categories and AffordanceNet-Context [83] as a state-of-the-art method for novel objects.

For an overview of the methods discussed see Table 2.3.

Method	**Year**	**Input**	**Output**	**Results**
S-HMP [76]	2015	RGB-D	Single-label	0.557 (UMD)
SRF [76]	2015	RGB-D	Single-label	0.466 (UMD)
Auto-encoder [80]	2016	RGB-D	Single-label	0.770 (UMD)
BB-CNN [39]	2017	RGB-D	Single-label	0.686 (IIT-AFF)
BB-CNN-CRF [39]	2017	RGB-D	Single-label	0.696 (IIT-AFF)
MultiNet [37]	2017	Depth	Single-label	None
AffordanceNet [12]	2018	RGB	Single-label	**0.799** (UMD) **0.734** (IIT-AFF)
MultiAffordanceNet [42]	2019	RG-D	Multi-label	0.62 (UMD) (Novel objects)
AffordanceNet-context [83]	2019	RG-D	Multi-label	0.69 (UMD) (Novel objects)
3D AffordanceNet [78]	2021	Point Cloud	Single-label	

Table 2.3: A table presents all results for the UMD and IIT-AFF dataset are given in the weighted F-measures as proposed by [79]

2.4 Handover evaluation

This section covers the considerations that need to be made when evaluating robot-to-human handover. There are no standardized set of test metrics and test methodology that is used consistently in human-to-robot handover.

2.4.1 Metrics

In a review from 2020, Ortenzi et al. [4] explored what metrics have been used to evaluate robotic handovers in previous work and seek to propose a set of standard test metrics. The review found three types of metrics used in robot handover. These were task-performance, subjective and psycho-physiological metrics. Success rate was found to be the most common task-performance metric but can be hard to compare across different testing methodologies if used as the only metric. In addition to task-performance metrics, a common approach is having subjective metrics in form of a post-questionnaire on a Likert scale. This is used to judge the participant's experience with the robot. Common questions are about robot predictability and how fluent the robot motion is. A set of standardized questions are proposed as part of the review. The questions can be found in appendix A. This questionnaire is only suitable if the test participant is not familiar with the robotic solution.

Additionally, psycho-physiological metrics, like EMG (electromyography) and HRV (heart rate variability) are metrics used to quantify the stress and activity of the test subject during the handover. Psycho-physiological metrics were not proposed in the review to be part of the standardised metrics as they were deemed too difficult to standardise when using sensors placed on the human body.
In addition to the proposed questionnaire, a set of task-performance metrics was proposed:

- Success rate
- Total handover time
- Receiver task completion time

The success rate of a robot handover is often just identified as the receiver got the object [84, 85, 86, 1]. This presents different challenges when comparing success rates as it is non-trivial to judge the quality of the handover. Here, the receiver task completion time and handover time are used to judge the quality of the handover. The total handover time can be an indicator of handover quality as it has been found that humans prefer a quick handover interaction [15]. Receiver task completion time seeks to identify if the received object orientation allows for completing a task without extensive in-hand manipulation of the object beforehand. If the grasp allows for a subsequent task to be easily performed, it should be considered more successful than one where the receiver is forced to make adjustments before starting [87].

2.4.2 Testing methodology

The choice of testing method should be such that the desired test metrics can be evaluated.

A testing methodology that seeks to test the effect of affordance grasping is performed by presenting an object to the subject in two different ways. One where the object is presented with the part intended for gripping to the subject and one where it is not. [7] This can be seen in Figure 2.7.

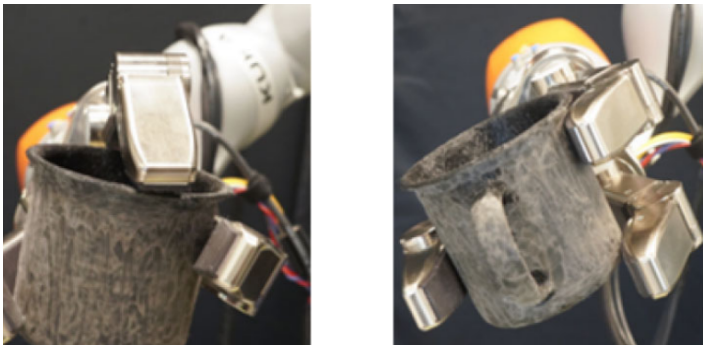


Figure 2.7: Handover where the cup is presented with its handle away from the subject (left) and toward the subject (right). [7]

The subject is asked to take the object from the robot, perform a task with the object e.g., filling the cup and then pressing a button. This is timed and compared to see how the completion time of a task is influenced by the way the object is presented to the user. The test methodology used is shown in Figure 2.8.

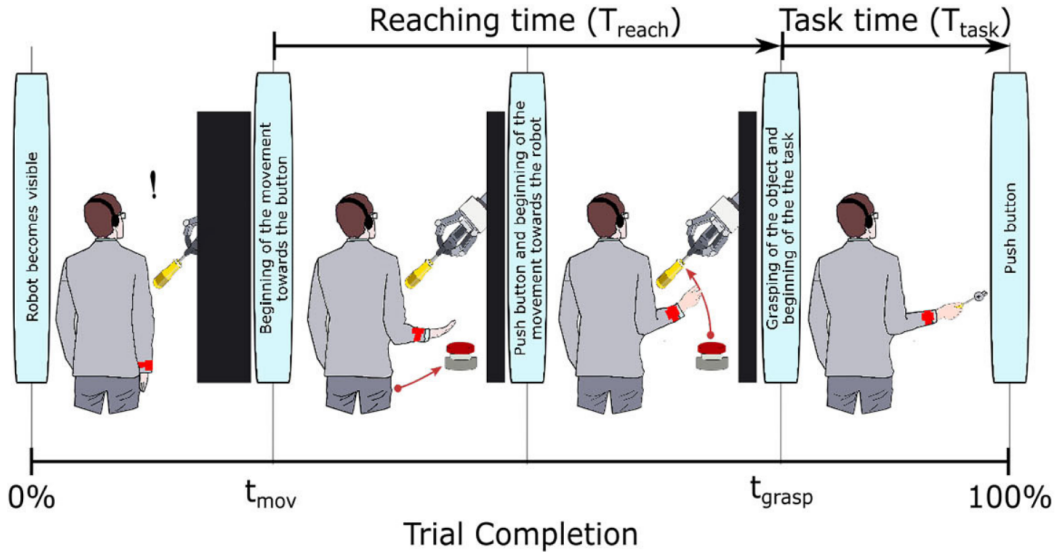


Figure 2.8: Handover test approach used for testing how task-oriented handover affects task completion time [7]

As an additional measure, the receiver's post-handover adjustment rate was used. It was recorded as no adjustment, in-hand adjustment, or bi-manual adjustment. Post handover adjustments rate is a metric used by several robotic handover solutions [4, 7]. It serves as an extra measure to evaluate handover quality. If the receiver can use the object straight away without adjustments, the receiver should be able to have a faster task completion time.
The proposed metrics of success rate, total handover time, task completion time, and post-handover adjustment rate will be considered for testing and evaluation of the developed solution.

2.5 Final problem formulation

The problem analysis gave the following insights into the task of task-oriented handover. According to [4], the handover can be split into two phases - a pre-handover phase and a physical exchange. Each phase comes with its own set of challenges to overcome. The pre-handover phase starts with initiation. The initiation can either be by object request or by task request. This projects only examines initiation by object request, where the receiver initate the handover by requesting a tool. Previous studies suggest that the percieved quality of the handover is increased if the robotic giver considers both the subsequent task of the receiver as well as the semantic constraints of the object. This is known as a task-oriented handover as discussed in Section 2.1.1.

For example, if the receiver asks for a hammer, the robot should grasp the hammer by its head rather than its handle, leaving the handle free for the receiver, this is known as task-oriented grasping. After grasping the hammer, it should move the tool to the handover location and orient the tool such that it can be comfortably grasped by the receiver. This sequence of actions requires the robot's software to autonomously synthesise a motion plan and a grasp that enables task-oriented handover.

After the tool is moved to the handover location, the tool should remain firmly grasped and stationary until the receiver reaches for it. Once the contact is made, the physical handover phase is initiated. During the physical handover phase, it is important to correctly modulate a grip force. If a force is too low, the tool might be released before the receiver has control over it. If a force is too high, the receive might need to pull the tool out of the robot's gripper, which could make the handover unsatisfactory.

Section 2.1.1 shows suggests that a task-oriented handover can be solved by combining state-of-the-art task-agnostic grasp synthesis methods with affordance segmentation methods. From Section 2.2 it was found that 6-DOF grasp synthesis methods are the best at grasping in occluded scenes. Furthermore, a method should be model-free in order to generalize to novel objects. To perform a fair evaluation, grasping should be evaluated with the YCB object set and the success rate metric.

Section 2.3 shows state-of-the-art performance in affordance segmentation is achieved with instance segmentation methods on known object. It is also possible to generalize af-

fordances to all objects by making the model object-class-agnostic, meaning it only detects background and foreground, but this method achieves slightly worse performance.

The focus of this report is on the pre-handover phase of a task-oriented handover. No further considerations will be made towards trajectory planning, as mentioned in Section 2.1.1. Additionally, handover location in the pre-handover phase and the physical handover phase is not considered further. Instead, a fixed handover location and orientation will be used, and the grip force modulation will be replaced with simple gripper control, that opens/closes the gripper.

With the problem analysis in mind, the final problem formulation for this report is stated as:

How can a UR5 cobot equipped with an Intel RealSense D435 and a Robotiq 3-finger gripper perform successful task-oriented handovers of objects from a custom object set by combining task-agnostic grasping methods with affordance segmentation?

2.5.1 Custom object set

In Section 2.2, the YCB object set was originally adopted for use in this report. However, the authors could not acquire this object set. Therefore, a custom object set is adopted. The custom dataset used in this report is a variation on the IIT-AFF object set [39], and it consists of a bottle, a bowl, a knife, a spatula, a hammer, and a cup. The object set is depicted in Figure 2.9.

Figure 2.9: The custom object set used for task-oriented handovers in this project.

2.6 Requirements

In order to assess the solution, each stage of the handover must be evaluated achieve certain performance requirements. The desired performance requirements will be discussed in this section.

2.6.1 Task-agnostic grasping

From the state-of-the-art (SOTA) analysis of task-agnostic grasp synthesis and grasping of objects, described in Section 2.2, it can be seen that the SOTA systems achieve between 82 % and 93 % grasping success. It is difficult to determine the best performing system as the systems vary in test setups, hardware and evaluation metrics. The set of objects that the various systems try to grasp also differs. Factoring in that the action of grasping a random part of an object is not the focus of this report, the authors deem a success rate of above 80 % as a satisfactory success criterion.

Requirement

- Success rate of task-agnostic grasping above 80 %

Where the success is defined as the ability to synthesize a valid grasp, performing grasping with the robotic system and moving the object to a pre-defined position without dropping the object. This is done on the custom object set as shown in Figure 2.9.

2.6.2 Affordance instance segmentation

In the field of affordance segmentation there exist clearly defined methods for evaluating the performance of object affordance segmentation models. AffordanceNet [12] achieved SOTA results on both the UMD dataset with an F_β^w score of 0.799 and a F_β^w score of 0.734 on the IIT-AFF dataset. With the next best method [80] achieving 0.77 F_β^w on the UMD dataset. The authors, therefore, strive to achieve an F_β^w of 0.75 or above.

While affordance segmentation is required to identify parts of an object that allow for a specific affordance, object detection is required to detect the different objects and tools. As such, the object detection performance of the proposed method also needs to be evaluated. The investigated methods in the affordance segmentation SOTA analysis did not report their object detection performance. Instead, other well known object detectors provide the success criteria. YOLO-v3 [88] and Mask-RCNN [82] are both evaluated on the COCO object detection dataset [89] which contains 80 classes. This is considerably more than the 10 objects categories present in the IIT-AFF dataset and the 17 object categories present in the UMD dataset. The Faster-RCNN object detector [90] achieved 78.8 % mean average precision (mAP) on the PASCAL VOC2007 dataset [91], which has 20 object classes in 9,963 images, making the results more comparable to the UMD and IIT-AFF datasets than

the results of the YOLO-v3 and Mask-RCNN. Faster-RCNN is no longer a SOTA method for object detection, but it is still widely used, and as such, sets a good object detection requirement.

Requirements

- $F_\beta^w > 0.75$
- mAP > 75 %

For an explanation of the F_β^w see equation (2.1) on page 19. The mAP score is computed with an intersection over union (IoU) threshold of 0.5 over each individual class as defined by the PASCAL-VOC2007 challenge [91].

2.6.3 Task-oriented grasping

Since task-oriented grasping is a subset of grasping, it suffers from many of the same issues as performance evaluation of grasping methods. There are no standardized datasets, nor an agreement upon selection of affordances to grasp. Moreover, the success criteria also differ. Due to these issues, it is difficult to compare the results of task-oriented grasping methods. As can be seen from the SOTA analysis in Section 2.1.1, the SOTA methods achieve between 69 % and 92.2 % task-oriented grasp success rates. The 69 % success rate achieved by Detry et al. [37] was tested by trying to grasp several different affordances. The work done by [80] and [39] primarily tried to grasp the *grasp* affordance with between 90 % and 92.2 % success rate. Since grasping only the *grasp* affordance is more limited than the aim of this report, the authors has decided on a required success rate closer to the work done by Detry et al.

Requirements

- Success rate of grasping the target affordance above 70 %.

Where the success is defined as having grasped the correct affordance and transporting it to a pre-defined position.

2.6.4 Task-oriented handover

In order to assess the performance of the task-oriented handover system, a baseline for comparison is needed. To the best of the authors' knowledge, there exist no reported results for which to compare a robot-to-human task-oriented handover system. In 2020, Rosenber et al. [1] proposed a system for human-to-robot handover that achieved a real life robotic success of 81.9 %. While this method solves a similar problem, it is not fully comparable. It is possible to instead look at some of the task-oriented grasping methods for

a baseline, as it involves many identical steps to achieve a similar objective. Detry et al [37], as previously discussed, achieved a 69 % success rate of grasping objects for a specific task, and then moving them to a position above a basket. Because the grasped object is moved from one pose to another, their work resembles this project in many way with the primary difference being the goal position, which is defined to be a handover location for this project. Chu et al. [42] also report the success of the task-oriented grasping followed by the manipulation of the object. They experimented with grasping object such as finding tools to scoop beans with. The physical manipulation experiment achieved the success rate of 82.5 %. Although they investigated a different problem, many of the same sub-problems had to be solved, such as task-oriented grasping and motion planning. Considering the two above-mentioned results, a success rate of 75 % seems to be reasonable. However, the performance of the task-oriented handover is tightly coupled with the task-oriented grasping. The success rate of the task-oriented grasping was set to 70 %, as defined earlier in this section. Moreover, the handover requires multiple modules cooperating, including trajectory generation, trajectory generation and initiation module. Therefore, the success rate of the task-oriented handover is set slightly lower than the success rate of the task-oriented grasping, leaving the success rate of the handover at 65 %.

Requirement

This report adopts the objective measure proposed by [4].

- Success rate of handover of 65 %

Where the success rate is defined as the system being able to grasp an object by the proper affordance and successfully transferring the object to the human receiver.

As was mentioned in Section 2.4, Ortenzi et al. [4] proposed two other objective measures in addition to the success rate. These measures are a total handover time and a receiver task completion time. However, because this project aims to propose a novel pipeline, thus focusing mainly on the technological aspects of the solution, the receiver task completion time or the total handover time will not be considered.

3 - Implementation

This chapter covers the implementation of the task-oriented handover solution. As was discussed in the problem analysis, see Chapter 2, the task-oriented handover consist of many different problems. In order to solve these problems, a solution must address each one of them. When designing the solution, considerations listed below were made.

- In order to capture information about objects that should be transported from the worktable to a human, a camera system is required.
- In order to grasp objects, grasp synthesis must be implemented.
- To perform task-oriented grasping, an affordance segmentation module must be implemented.
- With the affordance segmentation and the grasping synthesis being different modules, a third module that combines and associate both with each other should be implemented.
- A module that generate trajectory for the robot to follow must be made.
- In order to initiate the handover, the initiation module must be implemented.
- Finally, a main module that unifies the other modules and executes the task-oriented handover must be developed.

3.1 System overview

This section covers how the different modules of the system are interconnected and the hardware used. The details of each module implementation will be covered in separate sections. An overview of the full system can be seen in Figure 3.1.

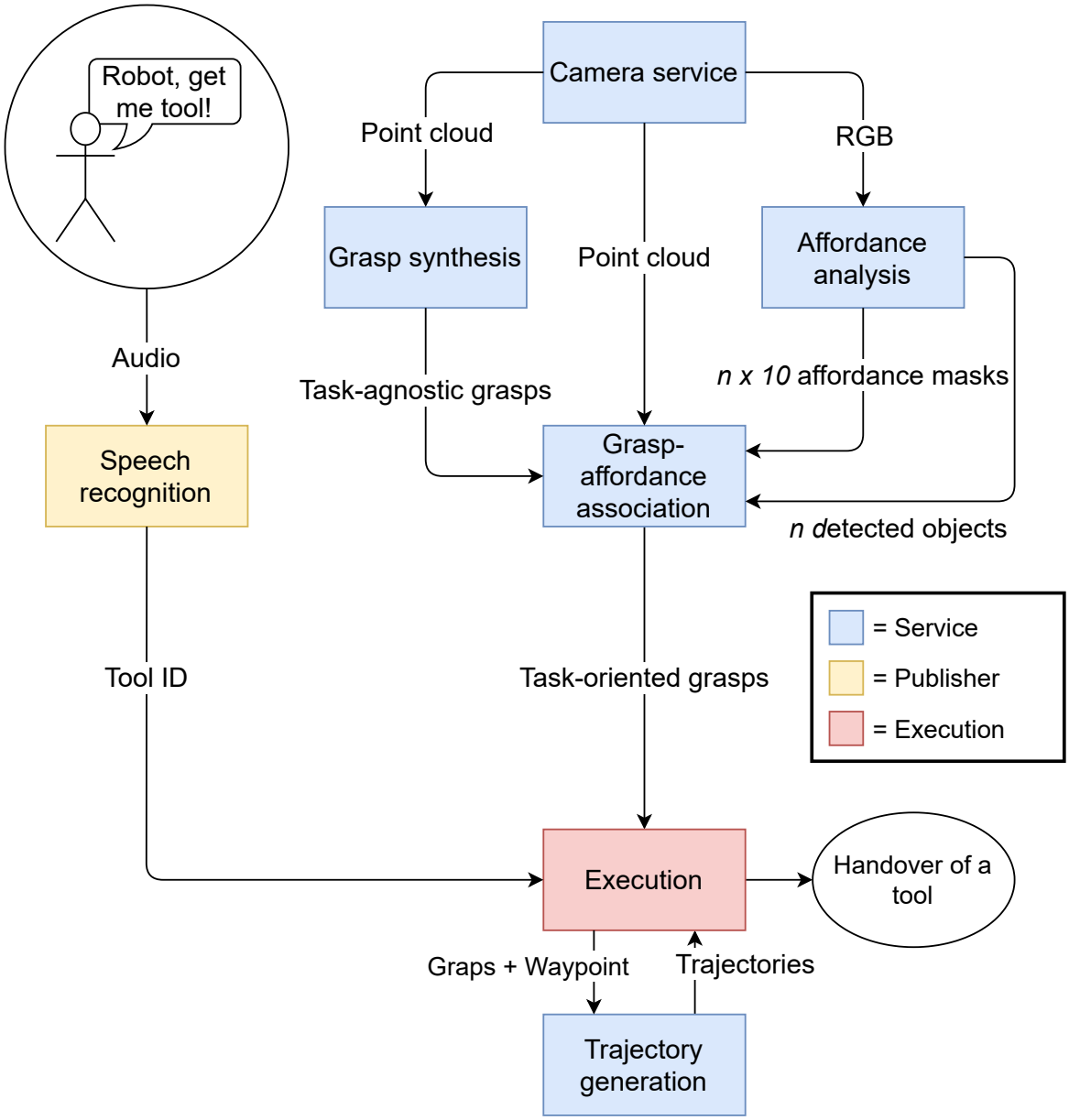


Figure 3.1: The system overview. Modules marked in yellow and blue are set up as ROS-publishers and ROS-services respectively. The module marked in red is responsible for execution of the handover.

- The *camera service* captures a new RGB-image and generates a new point cloud upon request. It ensures that all modules relying on this data get the same point cloud and image to work with.
- The *grasp synthesis* module generates a set of grasps for the point cloud provided by the *camera service*.

- The *affordance analysis* module generates 10 affordance masks and an object ID for each object using the RGB-image provided by the *camera service*.
- The *grasp-affordance association* module uses point cloud, grasps, affordance masks, and object IDs to associate affordance and grasps. Outputs grasps associated with a specific object and mask.
- The *trajectory generation* module receives grasps and waypoints from the *execution* and generates trajectory for the feasible waypoint and the grasp pair.
- The *speech recognition* module waits for an audio input. When audio is received, it recognises a requested tool. The tool is sent to *execution* as IDs.
- The *execution* receives IDs from *speech recognition* and trajectories from *trajectory generation*. A trajectory for the received object and affordance ID is executed to perform the handover.

The robot setup with the camera mounted on the pan-tilt unit(PTU) with a view of the work surface can be seen in Figure 3.2.

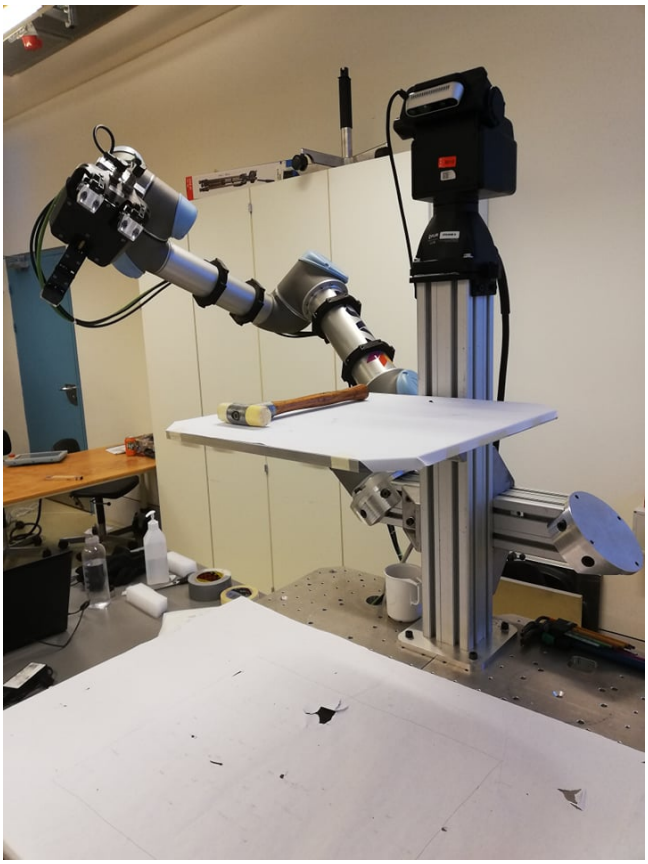


Figure 3.2: Picture of the robot setup, with a hammer placed on the work surface

3.1.1 Camera

The *camera service* module captures new information about the scene in the form of RGB and depth images, from which a point cloud is generated. The module interfaces with the RealSense D435 through the `pyrealsense2` library developed by Intel. The point cloud is generated through the `pyrealsense2` library and then post processed with a spatial filter and a temporal filter of size 10 collected at 5 hz. The spatial filter fills out holes in the point cloud by taking the distance closest to the camera, which favors reconstructing objects rather than the background. The camera operates in the high quality mode, with a resolution of the images is 1280×720. The *camera service* is, furthermore, in charge of synchronizing the input to the *affordance*, *grasp*, and *grasp-affordance association* modules, by keeping a pair of static synchronized RGB and point clouds, that can be requested by the other modules through a ROS service interface.

3.1.2 Pan-tilt unit

The PTU was controlled using the non-official ROS package [1]. The connection between the ROS master and the PTU is established wirelessly via the Transmission Control Protocol (TCP). The IP address of the PTU is 192.168.1.110. In order to move the PTU, `start_ptu` node should be launched. The node will then automatically connect to the PTU and tilts the camera to the desired angle, which was decided to be -63°.

3.1.3 Gripper

To connect to the robot's gripper, an official ROS package from Robotiq was used [2]. Same as with the PTU, the connection to the gripper is wireless and based on the TCP. The gripper's IP address is 192.168.1.140. In order to control the gripper, a control node `gripper_controller` was developed. This node takes as the input a single integer and translates it into more complex control messages for the gripper. The control node was developed on the foundation of the node `Robotiq3FGripperSimpleController` from the Robotiq ROS package. The gripper is used in the pinch mode, and it is controlled using open/close commands.

3.2 Task-agnostic grasp synthesis

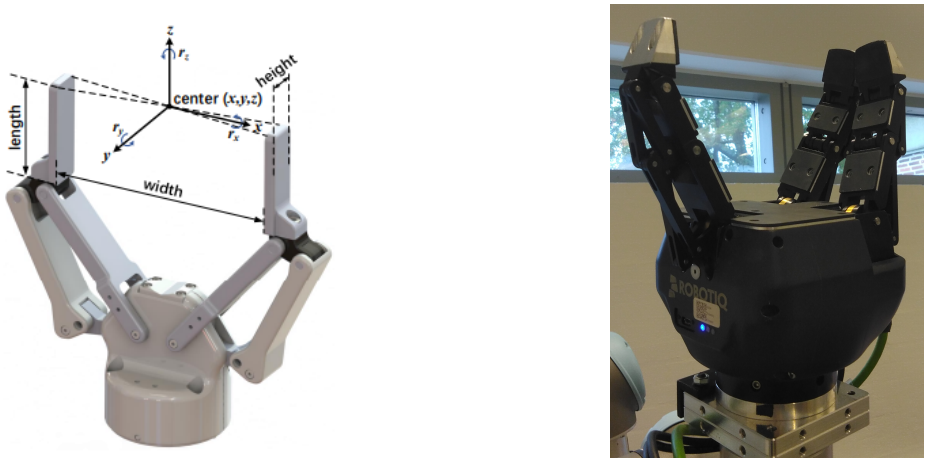
This section covers the grasping implementation of the developed handover solution. The purpose of the *grasp synthesis* module for this project is to generate task-agnostic grasps. Different possible grasps need to be generated to be able to choose a grasp that allows for handing over an object with the correct affordance, such as the handle of a tool, to the receiver. This module uses the point cloud generated by the *camera service*, such that the

[1] https://github.com/glhr/flir_ptu
[2] https://github.com/ros-industrial/robotiq

affordance analysis and *grasp synthesis* modules analyzes the same information. From this point cloud grasps are generated and outputted as ROS messages.

3.2.1 Approach

Grasp synthesis methods discussed in Section 2.2 were considered. GraspNet proposed by Fang et al. in 2020 [58] is chosen for its model-free approach that generates 6-DOF grasps for any object in the camera view. It has a publicly available pre-trained model on GitHub [3]. GraspNet achieves state-of-the-art performance on the GraspNet-1Billion dataset [92] created by the authors, where it outperforms previous SOTA solutions such as GG-CNN [59]. The GraspNet-1Billion dataset is captured using RealSense and Kinect cameras, and it is by far the largest publicly available grasping dataset and contains over one billion grasps. At the time of writing, the dataset is not widely used, as it was only made available in the middle of 2020. GraspNet is intended to be used with a parallel gripper with the gripper shown in Figure 3.3a



(a) (b)

Figure 3.3: (a) GraspNet gripper representation of a parallel gripper [53] [53]. **(b)** Robotiq 3-finger gripper set in pinch mode. Pinch mode allows the gripper to act as a parallel gripper.

To try and imitate a parallel gripper, the Robotiq 3-finger gripper was set to operate in pinch mode, where two fingers are close together to imitate a single finger of a parallel gripper. This finger configuration can be seen in Figure 3.3b. Using the Robotiq gripper in pinch mode could negatively impact the performance of grasping as the gripper's pinch mode is comparable to a parallel gripper but not exactly the same. However, initial observations showed that the grasps generated by GraspNet were graspable by the Robotiq 3-finger gripper in pinch mode.

[3] https://github.com/graspnet/graspnet-baseline

3.2.2 Method

GraspNet was developed by utilizing previous work such as Pointnet++ [74] with additions of newly proposed approaches by the authors such as grasp affinity fields [58]. An overview of the architecture of GraspNet, can be seen in Figure 3.4:

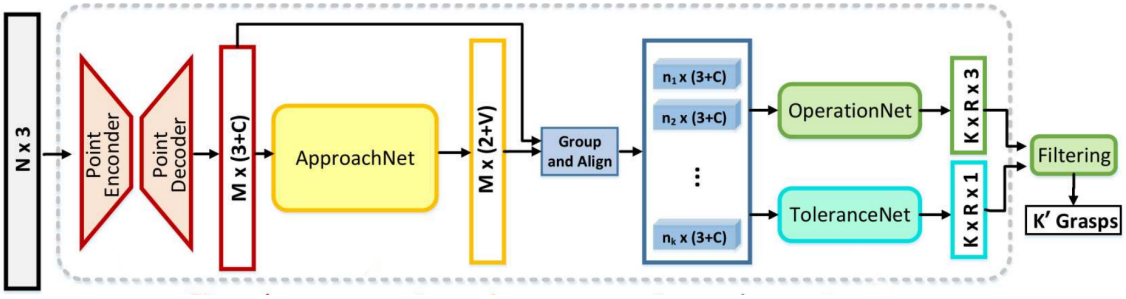


Figure 3.4: Overview of GraspNet architecture. Taken and modified from [58]

Base network

The network used as a base for GraspNet is Pointnet++. It handles extracting features from the point cloud. It takes a point cloud of $N \times 3$ points as input.

ApproachNet

The ApproachNet predicts whether a point is graspable or not. It finds approach vectors for the grasping point. These are classified to find which approach vectors are usable. The graspable or not graspable score along with usable approach vectors for each grasp is the output of the ApproachNet. [58]

Operation network

The operation network predicts approaching distance, gripper width, and grasp score. The grasp score is representative of how well the grasp should be able to pick up an object with a given grasp. Through experimentation by the authors of GraspNet, it was found that the grasp score is well correlated to real-world grasping performance [58]. This is shown in Table 3.1:

Object	s = 1	s = 0.5	s = 0.1
Banana	98 %	67 %	21 %
Peeler	95 %	59 %	9 %
Mug	96 %	62 %	12 %
Scissors	89 %	61 %	5 %

Table 3.1: Grasp success for different grasp scores(s). [58]

Tolerance network

The tolerance network uses the concept of grasp affinity fields introduced by the authors of GraspNet. It evaluates each grasp by determining how well the grasp can handle changes. This is done by looking at how far away adjacent grasps maintain a score of above 0.5. The larger distance you can move away from a grasp and still maintain a good score, the better the grasp is considered.

Finally, the output of the full GraspNet is a number of grasps for all objects in the point cloud. The output of GraspNet relevant for use in the project is the grasp pose (XYZ, rotation) and grasp score [58].

3.2.3 Implementation

The implementation of GraspNet for this project is implemented as a Docker image. The Dockerfiles are available on this project GitHub repository [93]. The *grasp synthesis* module is made available to other ROS nodes as a ROS service. This allows the *grasp-affordance association* to request a set of generated grasps whenever it is needed. It also allows the GraspNet to run on a separate machine. An overview of operations performed by the *grasp synthesis* module can be seen in Figure 3.5

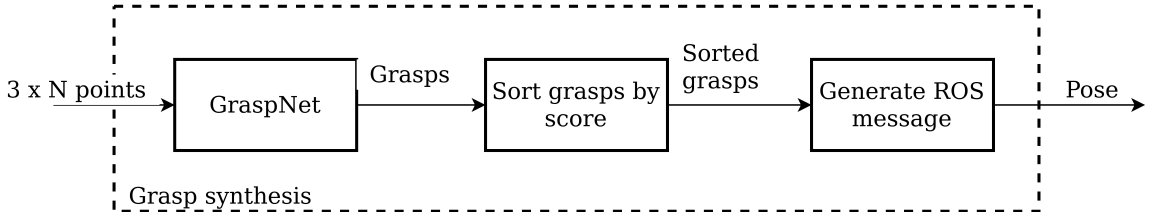


Figure 3.5: Example of grasps generated by GraspNet

The point cloud provided by the *camera service* module is used as the input for GraspNet. The set of grasps produces by GraspNet is sorted by score. The score is used in later modules to prioritize which grasps are processed first to find valid robot trajectories. Finally, a ROS message with the grasps position, rotation, and score is generated.

An example of grasps generated on a set of cluttered objects by GraspNet can be seen in Figure 3.6

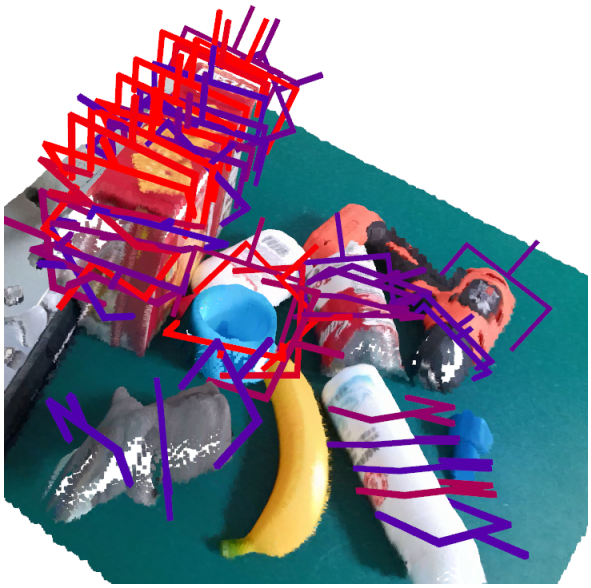


Figure 3.6: Example of grasps generated by GraspNet. The 50 highest scoring grasps are visualised.

All the grasp poses generated by GraspNet are sent together with their scores to be used in the *grasp-affordance association* module.

3.3 Affordance analysis

The *affordance analysis* module performs affordance instance segmentation of images captured from the RealSense D435 camera. The input is an RGB image and the output is n number of objects detected in the image along with $n \times 10$ affordance masks. This section explains how the module analyzes the input and produces the output.

3.3.1 Approach

From the SOTA analysis of affordance segmentation, see Section 2.3, it can be seen that AffordanceNet-Context proposed by Chu et al. [83] achieves the SOTA results on the UMD dataset for novel objects. However, initial tests of the AffordanceNet-Context's ability to detect objects shows that the network performs poorly with a mAP of 0.495.

AffordanceNet proposed by Do et al. in 2018 [12] achieved SOTA results on the UMD and IIT-AFF datasets for known objects and achieved a mAP 0.604 on detected objects, see Section 5.1. The authors of AffordanceNet provides an implementation with pre-trained weights[4], and it was decided to use AffordanceNet as the method for the *affordance analysis* module.

[4]https://github.com/nqanh/affordance-net

3.3.2 Method

The authors of AffordanceNet decided to cast object affordance segmentation as a traditional instance segmentation problem. This means that they detect objects and segments the affordance masks in parallel as one step. To this end, they use modified Mask-RCNN [82] which was the SOTA instance segmentation method in 2018. Nevertheless, Do et al's. problem definition was different to the problem definition that Mask-RCNN solves. Mask-RCNN performs object detection, classification, and segmentation. In the Mask-RCNN architecture, the segmentation mask is binary, that means each pixel is predicted to be either background or foreground. The authors of AffordanceNet aimed to segment each object into its various affordances, specifically 10 different affordances including a background class.

Mask-RCNN was, therefore, modified into AffordanceNet by changing the mask branch in the Mask-RCNN structure. Originally, the mask branch consist of two simple convolutional layers which produce a binary mask in parallel to the object detection and classification. This was modified by adding three blocks that each consist of a convolutional layer with ReLU activation, and then three deconvolution layers. The modified architecture is capable of outputting multi-class masks. The original Mask-RCNN architecture as well as AffordanceNet are depicted in Figure 3.7.

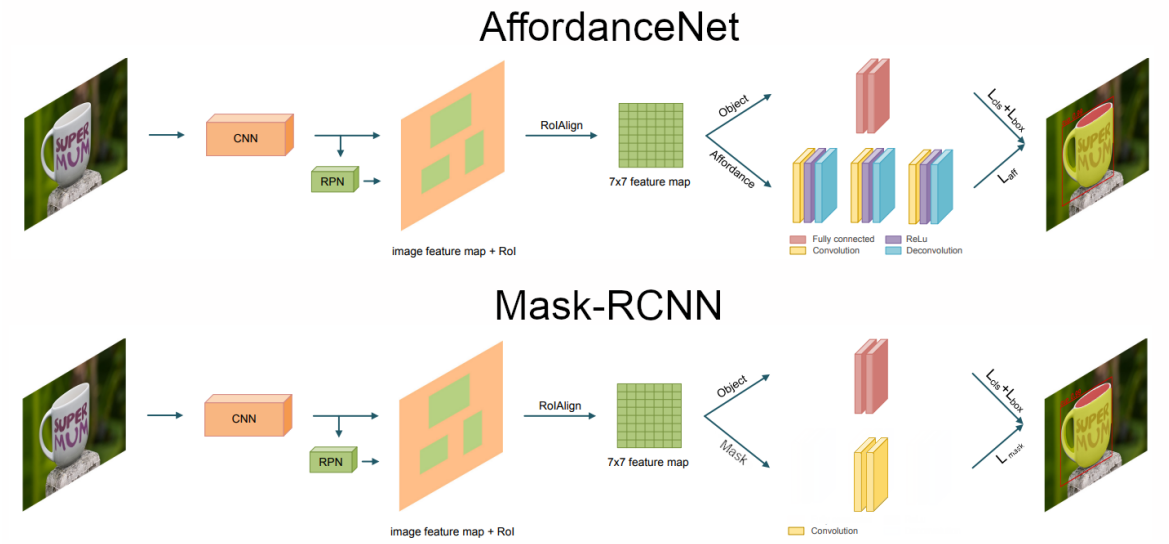


Figure 3.7: The architecture of AffordanceNet (top). Note that AffordanceNet has expanded upon the mask branch compared to Mask-RCNN (bottom). Adapted from [12] with information from [82].

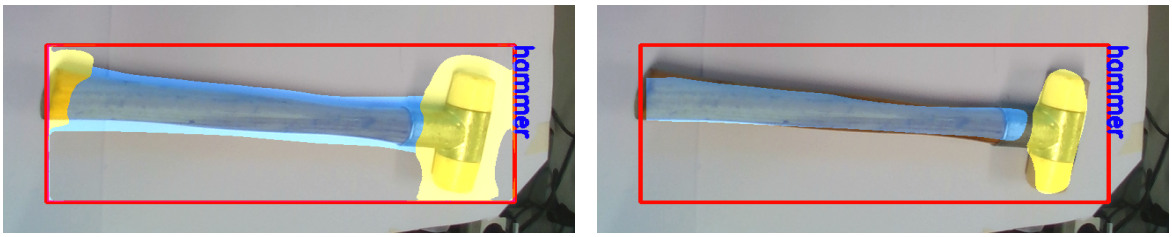
3.3.3 Implementation

For this project, the implementation provided by the original authors of AffordanceNet in the Caffe framework was implemented as a Docker image. The Dockerfiles can be found on the projects GitHub repository [93]. The implementation uses the pre-trained weights provided by the authors of AffordanceNet. The *affordance analysis* module is implemented as a ROS service which can be called by any ROS node in the system. This allows the *affordance analysis* module to run on a dedicated powerful machine.

The capabilities of AffordanceNet is determined by the dataset which it was trained on, which in this case is the IIT-AFF dataset, proposed by the authors themselves.

The system is capable of detecting the following 10 objects: bottle, bowl, cup, drill, hammer, knife, monitor, pan, racket, spatula. The system can segment the following 10 affordance classes: *contain*, *cut*, *display*, *engine*, *grasp*, *hit*, *pound*, *support*, and *wide-grasp*. The input to the network is an RGB image. The network is capable of analyzing an image of any resolution, but the resolution of 1280×720 pixels was chosen for this project. The output is 10 affordance masks for each detected object along with an object class.

Since the predicted masks suffers from some artifacts at the borders and affordance masks can be too big, the masks are post-processed with an erosion filter with size 13×13, as shown in Figure 3.8.



(a) **(b)**

Figure 3.8: Post-processing of AffordanceNet output. **(a)** A output of AffordanceNet before post-processing. Notice the oversized masks. **(b)** Affordance detection and segmentation after post-processing with a 13×13 erosion kernel.

3.4 Grasp-affordance association

This section covers how the results of the *grasp synthesis* module and the *affordance analysis* module are associated to allow for choosing grasps based on a desired affordance to pick up. An overview of the inputs for the grasp-affordance association can be seen in 3.9.

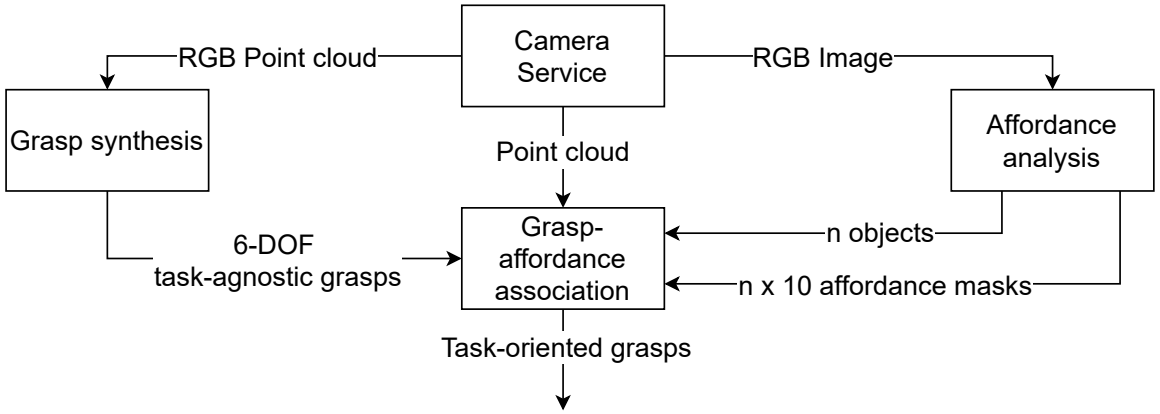


Figure 3.9: Overview of information flow in and out of the *grasp-affordance association* module

3.4.1 Input to the grasp-affordance association module

The *grasp-affordance association* module needs several inputs to be able to associate a grasp with a specific affordance. 6-DOF grasps are received from the *grasp synthesis* module as a pose (XYZ and quaternion) along with a score for each grasp. The same synchronized static point cloud that was used for grasp synthesis is also used here as the input. From the affordance analysis, a number of objects has been detected. For each object, 10 affordance masks are received along with a predicted class.

3.4.2 Method for associating grasps with affordances

The *grasp-affordance association* module associates the task-agnostic grasp with an affordance and an object class based on the masks produced by the *affordance analysis* module. Due to the nature of the 6-DOF grasps, the association is done in 3D space. First, the point cloud is segmented based on the affordance masks. Afterwards, irrelevant grasps are discarded based on a volume-of-interest (VOI). The remaining grasps are extrapolated to estimate whether they will grasp the affordance of interest. The algorithm is visualized in Figure 3.10. This section explains the algorithm in depth.

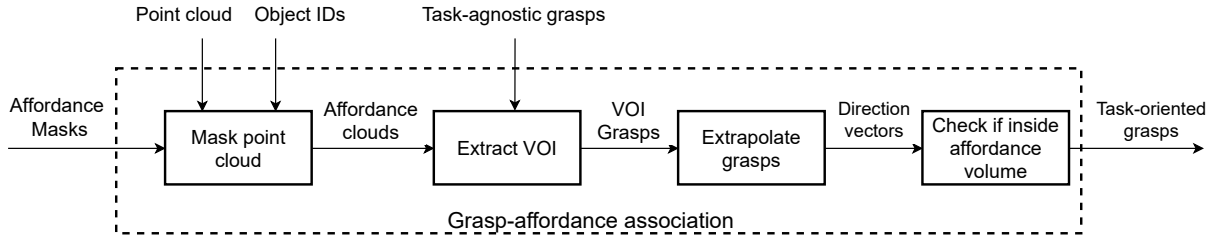


Figure 3.10: Flowchart of grasp-affordance association.

Masking the point cloud

To be able to associate specific grasps with the mask, the mask pixels needs to be associated with points in the point cloud.

The *grasp synthesis* and *affordance analysis* module produces grasps and affordance segmentation masks in the same camera frame. Therefore, identifying the points in the point cloud that belong to a given affordance mask is straightforward. The RealSense D435 depth camera is supported by the `pyrealsense2` library developed by Intel. This library has built-in functions that can identify which pixels in the RGB image belongs to which points in the point cloud. The affordance masks can, therefore, be used to select the points in the point cloud that are associated with a given affordance, see Figure 3.11.

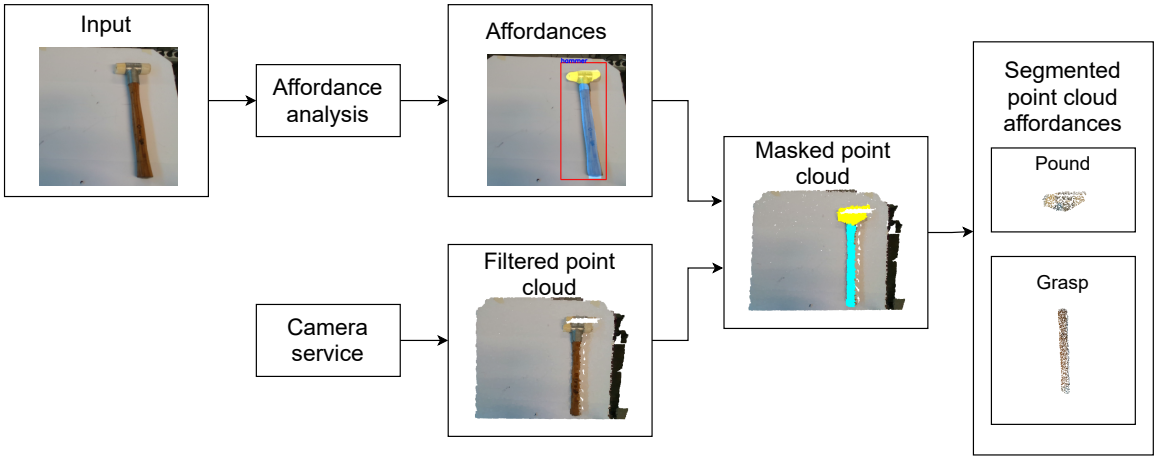


Figure 3.11: The points associated with the different affordances can be found from the predicted affordance masks in the RGB image.

Extract volume of interest

In order not to examine every grasp produced by the *grasp synthseis* module, some of the grasps are discarded by checking if they belong to a VOI.

The VOI is found by constructing a bounding box around the masked point cloud. Outliers in the masked point cloud are removed if they are further than 2 standard deviations measured in euclidean distance from the nearest 10 neighbours. This is done before computing the bounding box. The bounding box is defined by 2 points, P_{min} and P_{max} in 3D space. The VOI is enlargened 10 cm in all directions. The points are defined in equation (3.1) and (3.2).

$$P_{min} = \begin{bmatrix} min(V_x) - 10cm \\ min(V_y) - 10cm \\ min(V_z) - 10cm \end{bmatrix} \tag{3.1}$$

$$P_{max} = \begin{bmatrix} max(V_x) + 10cm \\ max(V_y) + 10cm \\ max(V_z) + 10cm \end{bmatrix} \tag{3.2}$$

Where V is the VOI, and $min()$ and $max()$ outputs the minimum and maximum value of the different x, y, z axis.

All grasps that belongs to the VOI are considered as nearby grasps and are potential candidates for grasping the affordance. The process is visualized in Figure 3.12

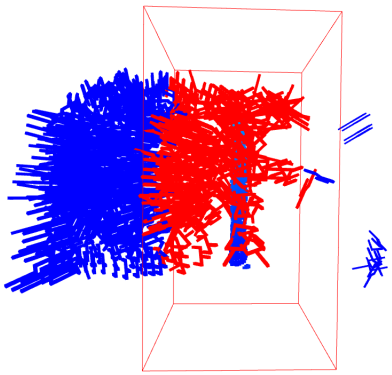


Figure 3.12: All grasps positioned within the VOI are depicted in red, and the ones outside are depicted in blue. Only the red grasps will be further examined.

Extrapolation of grasps

Each grasp has a pose and an orientation. This information is used to extrapolate the direction for which the grasp is facing. A direction is used to check whether it belongs to a given affordance.

A direction vector d is computed from the quaternion of the grasp q by rotating a unit vector u in the x-direction and making it a quaternion as defined in (3.3), and rotated as in (3.4). The direction vector is defined in equation (3.5)

$$u = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.3}$$

$$p = q \times u \times q^* \tag{3.4}$$

$$d = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{3.5}$$

The direction vector d is then extrapolated 10 cm with a discrete resolution of 0.5 cm. From here it can be checked if the grasp is oriented towards the affordance point cloud.

Associate grasp and object

Another bounding box is constructed around the masked point cloud, unlike the first bounding box referred to as a VOI, this one is non-axis aligned. This means that the bounding box is made to be as small as possible, while still encapsulating all points in the masked point cloud.

In order to check if a grasp belongs to an affordance, it is checked whether any of the 20 discrete extrapolated points of the direction vectors lies inside the non-axis aligned bounding box. In practice, this is done by examining if the distance from each of the extrapolated points to the center of the non-axis aligned bounding box violates the boundary of the bounding box. This is shown in Figure 3.13.

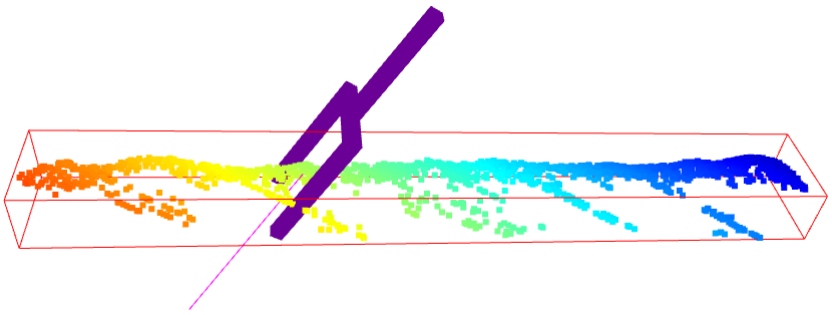


Figure 3.13: A grasp, depicted in dark purple, is examined to see whether it will grasp the masked point cloud. A part of the extrapolated line, depicted in bright purple, is within the bounding box. This grasp is therefore predicted to grasp the masked point cloud.

If any of the extrapolated points belongs to the bounding box, the associated grasp is considered to be associated with the affordance belonging to the masked point cloud.

With this method, all grasps can be associated with an affordance as segmented by the *affordance analysis* module. An example of the input and output to this module is shown in Figure 3.14.

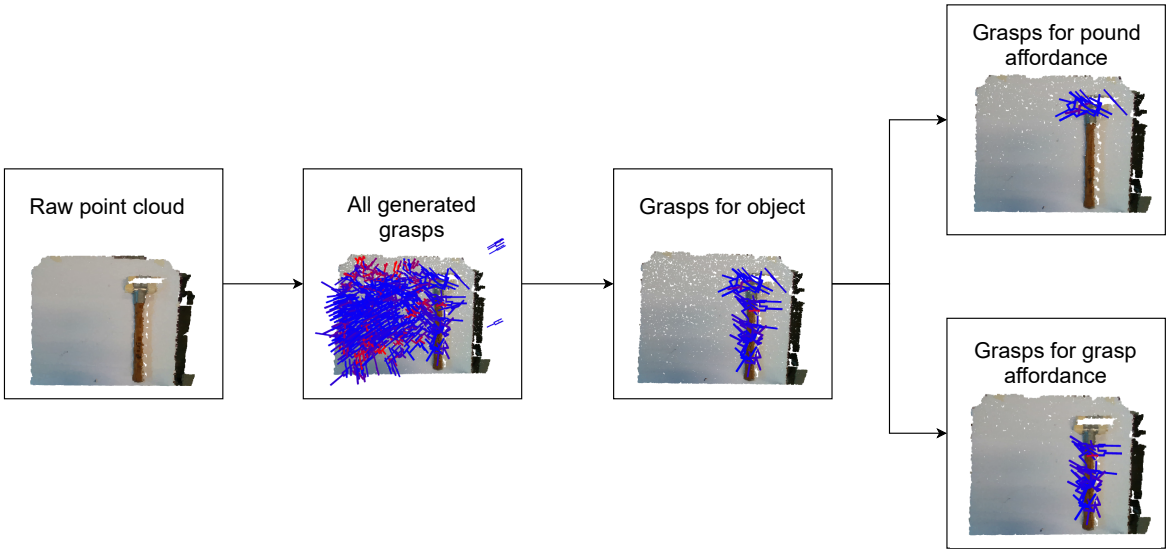


Figure 3.14: Given a point cloud with a set of grasps generated by the *grasp synthesis* module, grasps can either be associated with individual objects or an affordance. In this example, the grasps for a hammer is found.

Finally, the set of remaining grasps are published. These will be used to generate trajectories and execute grasps for a requested object.

3.5 Speech recognition

This section covers how speech recognition is performed to execute the pickup of specific objects. In the developed handover pipeline, the speech recognition acts as the initiation of the handover. It allows the human receiver to start the handover by requesting a specific object and the object affordance to be targeted. An overview of the speech recognition can be seen in Figure 3.15.

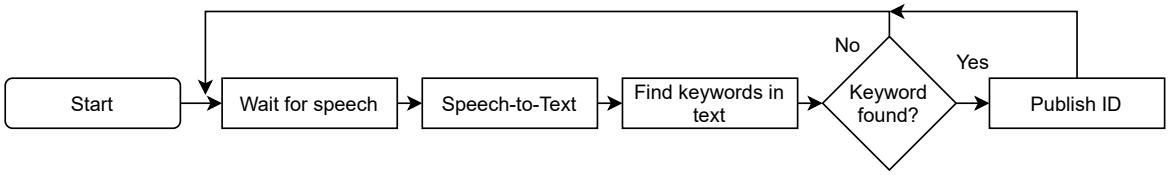


Figure 3.15: Overview of the *speech recognition* module

The node will then wait for the audio input. It records a sentence up to 5 seconds long and stores it as audio that can be used for speech-to-text. The speech-to-text is performed with the `SpeechRecognition` library as an interface for using Google Speech Recognition [94]. The text output of the speech-to-text is searched to find if any of the desired keywords are present. The desired keyword is either a tool (hammer, knife, etc.)

or an affordance to target. This procedure is done twice. First, the receiver will ask the robot to pick up an object. If the object is part of the object keywords, the receiver can then ask for a specific affordance for the object to be picked up by. If a valid affordance is named in the sentence, the node will publish both the object ID. The objects affordances and their IDs match the objects detectable by AffordanceNet. Additionally, the two extra keywords, tool and handle, are added. Handle refers to the part of the object with as *grasp* or *wide-grasp*, and tool refers to the *functional* affordances. If no keywords or keywords for multiple different objects or affordances are found in a sentence, no ID will be published and the node will wait for new speech input.

The *speech recognition* module was developed and tested to be able to identify the desired object followed by the desired affordance. For the final solution, only a desired object is required for initiation, as the target affordance is always the *functional* affordance as the aim of the task-oriented handover is to present the operator with the tool handle. The speech recognition allows for easy initiation of the handover by the receiver. The IDs published by the speech recognition allow the robot to start executing a grasp for a specific affordance of a named object.

3.5.1 Testing speech recognition

A test of the standalone speech recognition is performed to evaluate if it performs sufficiently well to comply with the project requirements presented in Section 2.6.

The test is performed with a test participant from the authors. The participant is asked to say the following: *"Robot pick up the **object**"* where ***object*** is one of the test objects:

- a hammer,
- a spatula,
- a knife.

Then, the participant will say: *"Pick up by the **affordance**"* where ***affordance*** is either a tool or a handle. The test is performed for each object affordance combination 10 times. Testing is done with a headset microphone. The background noise in the environment is low, and no other people are talking during the test. The result will be counted as a success if the speech recognition publishes the IDs corresponding to the object and affordance that was asked for. The results of the test can be seen in Table 3.2

Object	Hammer		Knife		Spatula	
Affordance	Handle	Tool	Handle	Tool	Handle	Tool
Success	90 %	90 %	100 %	90 %	80 %	90 %
Failure	10 %	10 %	0 %	10 %	20 %	10 %

Table 3.2: Results of speech recognition test.

The results show an overall 90% success rate of speech recognition. The failures recorded were all caused by the text-to-speech not recognising what was said. A higher success rate would be preferential, but in all cases, either the correct or no ID was published. The testing shows that the robot will be able to go for the correct object. As such, the current solution to no object ID being found is simply asking for the object again.
This speech recognition implementation is deemed sufficiently successful to act as the initiation of the handover and be a part of the developed pipeline. The following Section 3.6 will cover the execution and trajectory generation.

3.6 Execution and trajectory generation modules

In order to move the robot from its starting pose to the desired grasping pose, a trajectory must be planned. As was stated in Section 2.1.1, the ROS-native MoveIt is used as the motion planning and execution framework. This section describes the implementation of both trajectory generation and motion execution.

3.6.1 MoveIt preparation and requirements

In order to control the robot using MoveIt, configuration files had to be generated. To do this, a URDF file of the robot setup must be provided. The written URDF can be seen in Figure 3.16. Both the URDF and the MoveIt configuration files can be found on the project's GitHub repository [93].

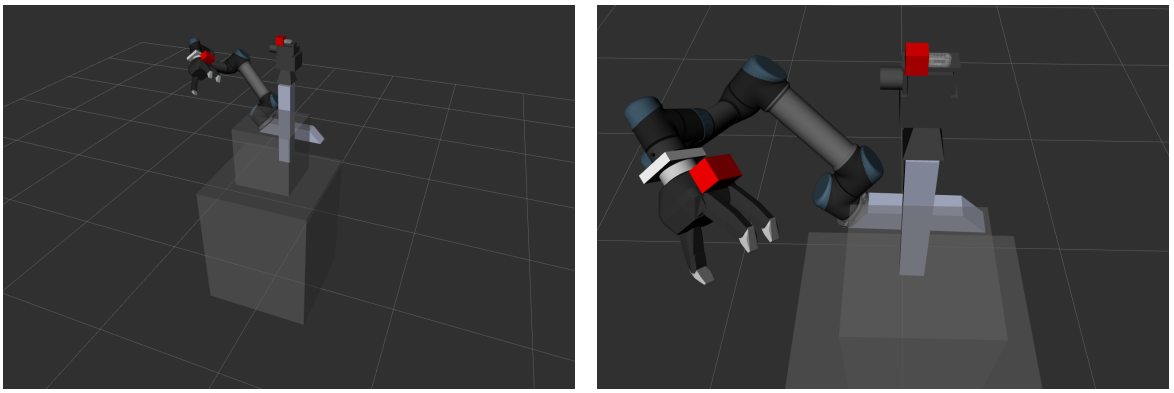


Figure 3.16: The URDF of the robot setup. Note the red rectangles, which were inserted into the URDF to protect the areas where the cables are connected but are not physically in the robot setup.

3.6.2 Execution module

The *execution* module calls the other developed modules and at the end executes the robot's motion. First, the grasps are generated via the *grasp synthesis* module. Then, the affordances are identified with the *affordance analysis* module. Afterwards, the grasps and

affordances are associated through the *grasp-affordance association* module. The association process uses the tool ID derived from the user's input using the *speech recognition* module. The grasps that do not belong to the desired tool or the tool's *functional* affordance are removed.

Next, a pre-grasp pose is calculated for each remaining grasp. The pre-grasp shares the orientation with its grasps, but it is positioned -10 cm along the x-axis of the grasp. An example of the pre-grasp pose and the grasp pose can be seen in Figure 3.17. By moving to the pre-grasp pose before the goal grasp configuration, the grasp pose can be reached in a linear motion. If the robot was to move to the grasp pose directly, skipping the pre-grasp pose, it may collide with the object, effectively moving the object, before attempting to grasp it. Because the object was moved, the grasp as received from the *grasp-affordance* module is no longer valid, therefore the grasping may fail. The collision with the object happens because the *trajectory generation* module does not account for the point cloud input from the *camera service* module, and therefore cannot plan around the object. The list of grasps and pre-grasps is used as input to the *trajectory generation* module. The *trajectory generation* module is described in Section 3.6.3.

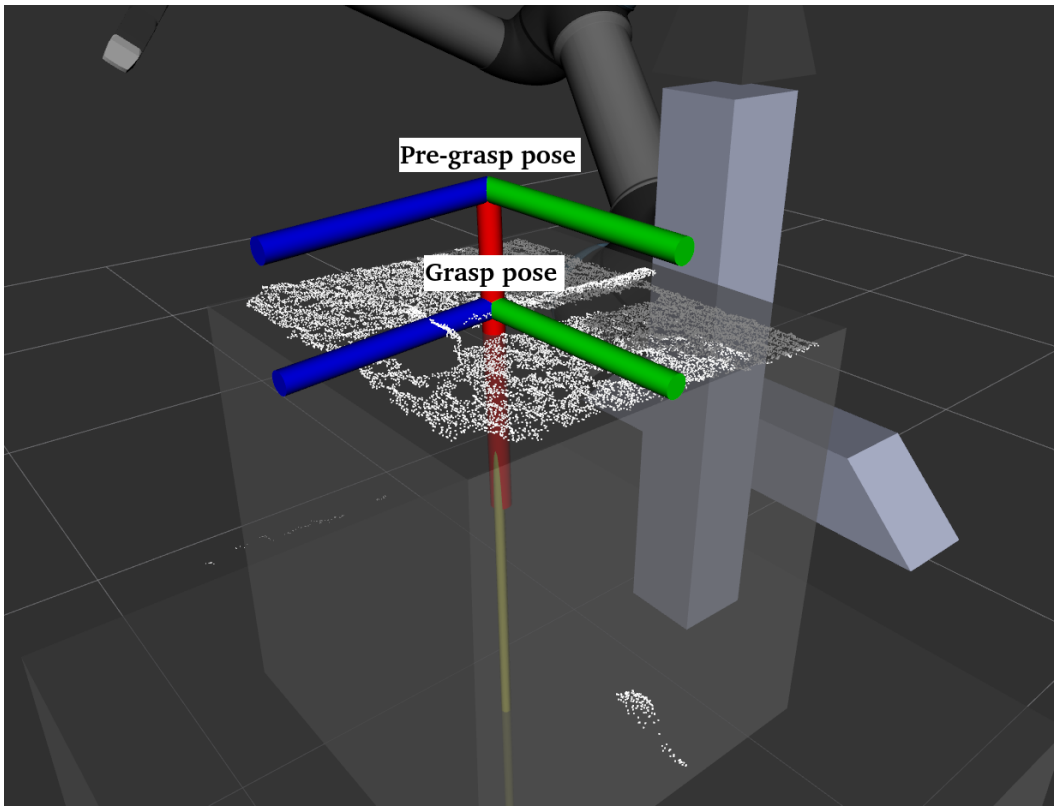


Figure 3.17: The visualisation of the pre-grasp pose positioned -10 cm along the X-axis of the grasp pose.

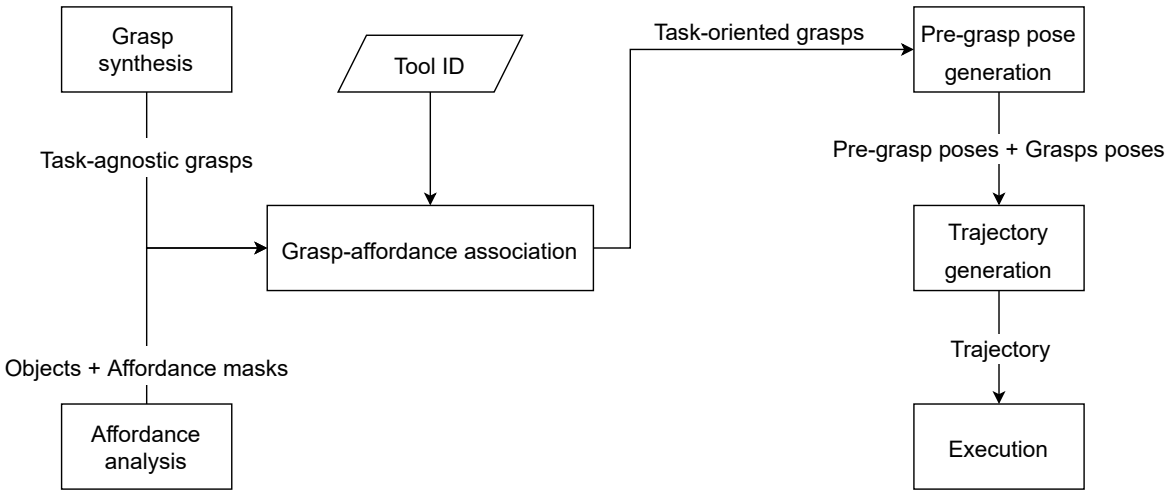


Figure 3.18: The block diagram of the execution module.

After the trajectories are returned to the *execution* module, the trajectories are sequentially executed. The sequence, visualised in Figure 3.19, is as follows:

1. Go to pre-grasp pose.
2. Go to grasp.
3. Go back to the pre-grasp pose.
4. Go to home pose.
5. Go to handover pose.
6. Go back to the home pose.

The home pose and the handover pose were predefined during the MoveIt setup. Lastly, the system is open-loop, which means that the failure to grasp is not registered and the robot will not attempt to re-grasp.

The Figure 3.18 visualises the overview of the described *execution* module.

3.6.3 Trajectory generation module

The input to the *trajectory generation* module is the list of grasps and pre-grasps poses. The list of grasps is traversed until a feasible trajectory is found. The feasibility of both the grasp pose and its associated pre-grasp pose is checked via the inverse kinematics solver. If the inverse kinematics solver validates the pre-grasp pose, it will move on to validate the grasp. A pose is validated if the inverse kinematic solver can find a valid joint configuration corresponding to the pose. If both poses are feasible, the joint angles at the pre-grasp pose and the grasp pose, as calculated by the kinematics solver, are used as

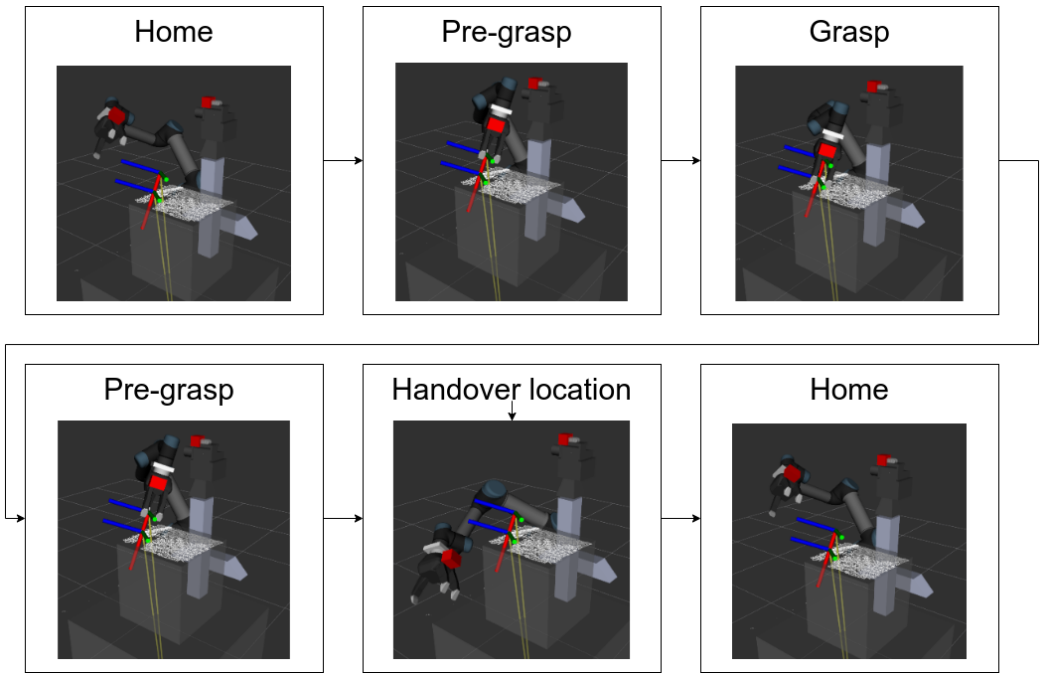


Figure 3.19: The visualisation of the motion sequence robot performs for each handover.

the goal configurations for trajectory planning. If the inverse kinematics solver does not validate the pre-grasp pose or the grasp pose, no trajectory is generated. The overview of the proposed algorithm can be seen in Figure 3.20.

During the trajectory planning phase, the robot's state space is explored using the RRTConnect algorithm as implemented in the OMPL MoveIt planner library. RRTConnect is a variation of the rapidly-exploring random tree (RRT) algorithm. RRT traverses the state space using a single tree, which expands from the initial robot's state *q_start*, until the path to the goal configuration *q_goal* is found [95]. RRTConnect explores the joint space using two trees - one being generated from *q_start* and the other from *q_goal*. When the two trees connect, the trajectory from *q_start* to *q_goal* is found [96].

The set of trajectories produced by the module consists of:

- A trajectory from the robot's initial pose to a pre-grasp pose,
- A trajectory from the pre-grasp pose to a grasp pose,
- A trajectory from the grasp pose back to the pre-grasp pose.

The last trajectory, from the grasp pose to the pre-grasp pose, is used to safely move the item from the work table towards the user. Without moving back to the pre-grasp pose, the robot, that is unaware of the grasped item at its end-effector, might move in a way that causes a collision between the table and the item, which could result in a drop of the item. This trajectory is generated anew rather than reversed from the pre-grasp pose to a grasp

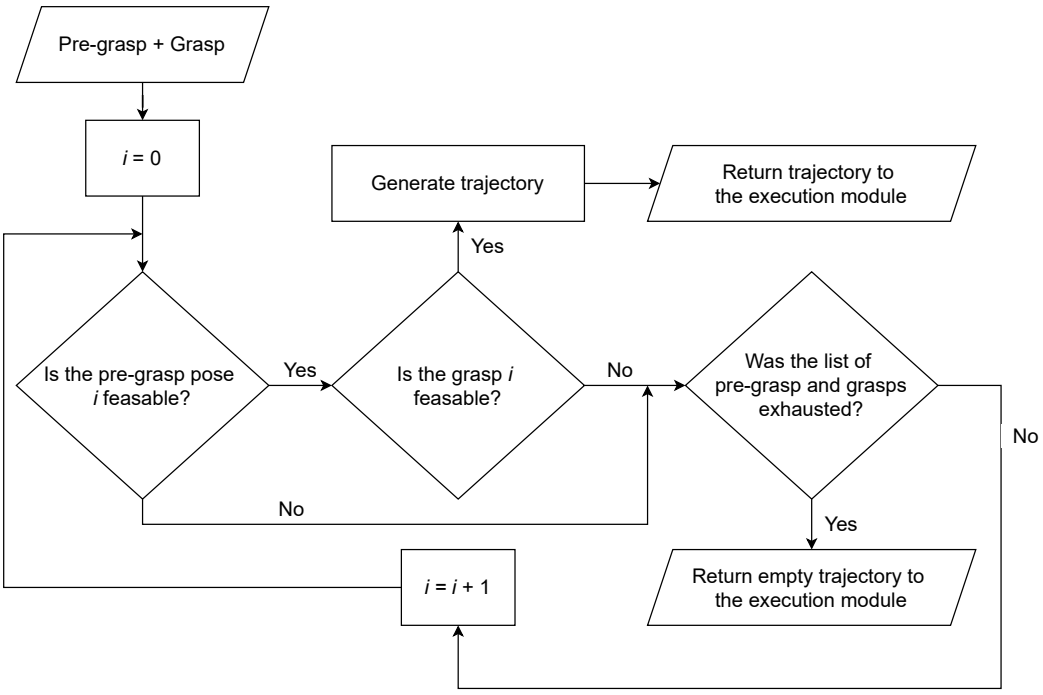


Figure 3.20: The flowchart of the trajectory generation module.

trajectory. Finally, the robot's initial pose is constant for each grasp. This pose is set to be the robot's home pose, predefined during the MoveIt setup.

The selection of the kinematic solver

Initially, the KDL kinematic solver, a default MoveIt option, was used. However, the performance was unreliable, with the solver often failing to find an inverse kinematics solution. Therefore, a different kinematic solver was chosen. The BioIK solver proved to be a fine choice. In the introductory paper for the solver, the solver regularly outperforms the KDL solver and matches the performance of the TRAC-IK solver [97]. However, when tested on the robot, the BioIK solver proved to be a better choice than TRAC-IK. Details on how the BioIK solver works can be found in its introductory paper [97].

4 - Testing

This section describes the test setup and a procedure for each test carried out for the proposed solution.

4.1 Affordance object detection

Since the *affordance analysis* module consist of AffordanceNet [12], with pre-trained weights provided by the authors of AffordanceNet, the authors of this report assumes that the *affordance analysis* module performs similarly in terms of F_β^w score. Therefore, the F_β^w score is not tested. Since the *affordance analysis* module utilizes AffordanceNet which report a F_β^w of 0.79 it is assumed that the *affordance analysis* module fulfills the F_β^w requirement.

The purpose of this test is to evaluate the performance of the object detection of the *affordance analysis* module as the rest of the handover system can not function if the object detection fails.

Information flow

For this test, a dataset consisting of 30 images with a total of 113 objects was annotated. The images for the dataset were captured from the RealSense D435 camera mounted on top of the Little Helper v7 system angled at -63 degrees. 30 images with different combinations of objects from the object set described in Section 2.5 were present. Each object in the 30 images was annotated with a bounding box and an object ID. The dataset serves to act as a ground truth to subsequent predictions performed by the *affordance analysis* module. The dataset is representative of data the system will encounter, and a subset of the dataset can be seen in Figure 4.1.

The two inputs to the test are the RGB images from the dataset, which is analyzed by the *affordance analysis* module, and the ground truth from the annotated dataset. The setup is visualized Figure 4.2.

Metrics

- Mean Average Precision (mAP)

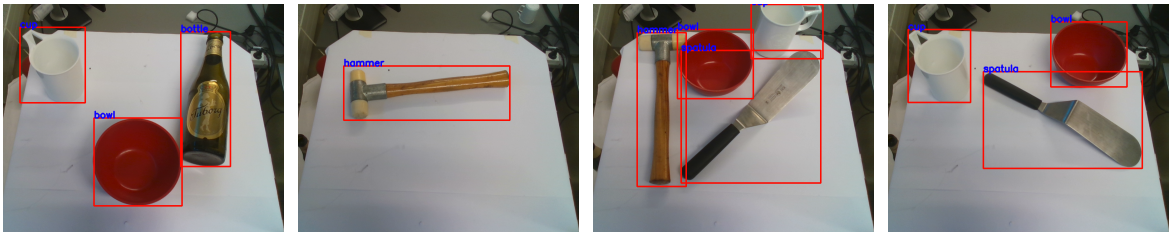


Figure 4.1: Sample images from the dataset consisting of annotated bounding boxes in 30 images captured from the RealSense camera.

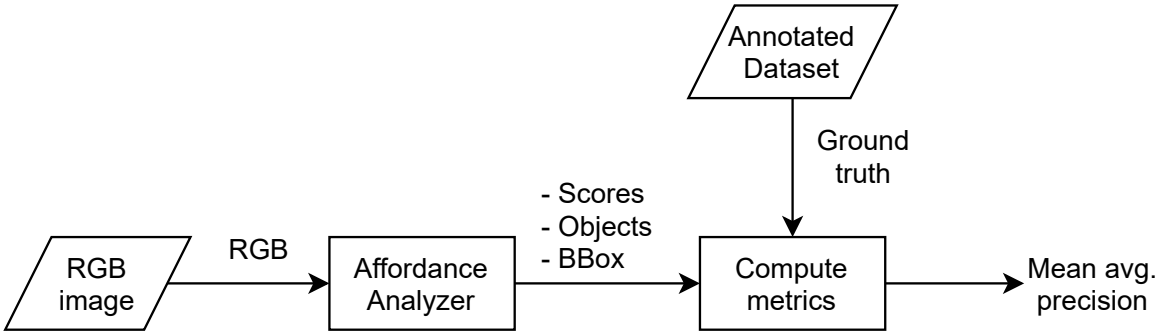


Figure 4.2: All of the modules involved in the affordance test with a visualization of how information flows between the modules.

As mentioned previously in Section 2.6, the system requirement is based on the object detector Faster-RCNN [90], which report the performance in mAP with an IoU threshold of 50 %.

Procedure

The following procedure for the affordance object detection was developed:

1. Predict *score*, *object* and *bounding box* with the *affordance analysis* module for each image in the dataset.
2. Using the ground truth, compute the mAP.

The test is programmed and available as a script on the project's GitHub repository [93]. The results of the affordance object detection test can be seen in Section 5.1.

4.2 Task-agnostic grasping test

The performance of the grasping pipeline was evaluated with the test described in this section. This test only evaluates the *grasp synthesis* and *trajectory generation* modules, with the *affordance analysis* module being only partially involved.

Information flow

From the RealSense camera, an RGB image of the environment is captured and a point cloud of the environment is constructed. The RGB image is analyzed with the *affordance analysis* module, which computes a mask for each object present. The point cloud is processed as described in Section 3.1.1 and analyzed with the *grasp synthesis* module. The *grasp synthesis* module generates a set of grasp for the entire environment captured by the RealSense camera. Afterwards, the grasps that can grasp the object present in the environment have to be found. The grasp-object association is described in Section 3.4. The resulting grasps are then sorted according to their score, as computed by the *grasp synthesis* module, and fed into the *trajectory generation* module which generates a trajectory for each proposed grasp. MoveIt is then used to execute the resulting trajectory. The full flow of information can be seen in Figure 4.3.

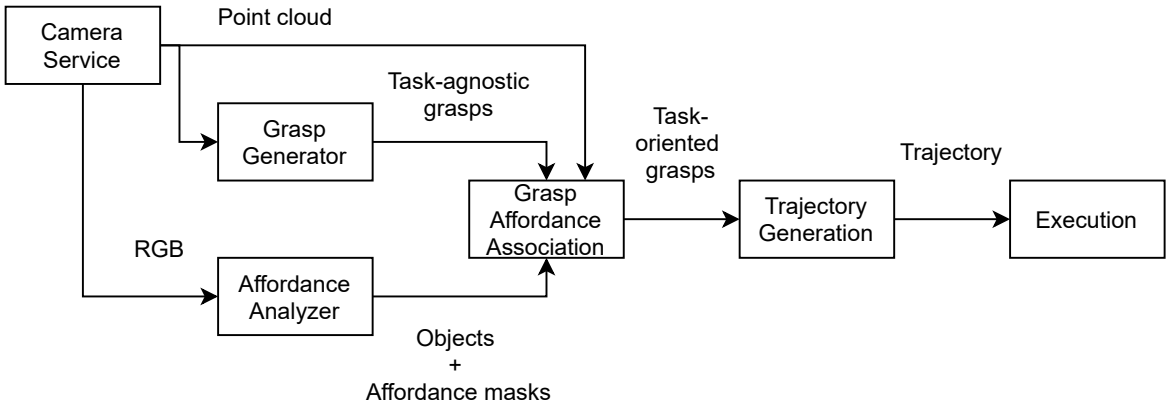


Figure 4.3: All of the modules involved in the grasping test with a visualization of how information flows between the modules.

Metrics

The grasping capabilities of the proposed grasping pipeline are evaluated with the following metrics:

- Success: Binary, the robot was able to grasp the intended object and transport it to a predefined pose without encountering any failures.
- Failure to generate grasps: Binary, the *grasp synthesis* module could not produce any grasps or failed to generate trajectory for any of the suggested grasps. Failure to generate trajectory is assumed to be failure to generate grasps because the *grasp synthesis* module failed to produce feasible grasps.
- Failure to grasp: Binary, the robot executed a grasp but failed to grasp and stabilise the object, thus not grasping the object.

- Failure to grasp and hold: Binary, the robot succeeded in grasping the object, but dropped it during transportation to a predefined pose.

Procedure

The grasping pipeline was evaluated on the object set described in Section 2.5, consisting of the following items: a hammer,a knife, a cup, a bowl, a bottle, and a spatula.

The following is the procedure for the grasping test.

1. Place an object from the object set in front of the robot and fully in the field of view of the camera.
2. Confirm that the *affordance analysis* module was capable of detecting and segmenting the object. If not, return to step 1.
3. Execute the program.
4. Repeat 5 times for each object.

Following the test procedure results in a total of 30 tests performed of the task-agnostic grasping. Although this test aims to only evaluate the grasping capabilities, step 2 of the procedure is essential. Without it, it is not possible to filter out the grasps that do not belong to the object being tested, such as grasps generated for the worktable itself. The *affordance analysis* module itself is tested separately.

The test is programmed and available as a script on the project's GitHub repository [93]. The results of the grasping test can be seen in Section 5.2.

4.3 Task-oriented grasping

The proposed method of task-oriented grasping was evaluated by measuring the system's ability to target a specific object affordance when grasping an object.

Information flow

The flow of information in the task-oriented grasping test is identical to the flow of information in the grasping test, see Figure 4.3, with the exception of the output of the *grasp-affordance association* module. Instead of associating grasps with all the affordances of an object as in the grasping test, trajectories are generated for grasps with either the *grasp* affordance or the *functional* affordance, as shown in Figure 4.4. The *functional* affordance consist of all grasps that are not labelled with the *grasp* or *wide-grasp* affordance.

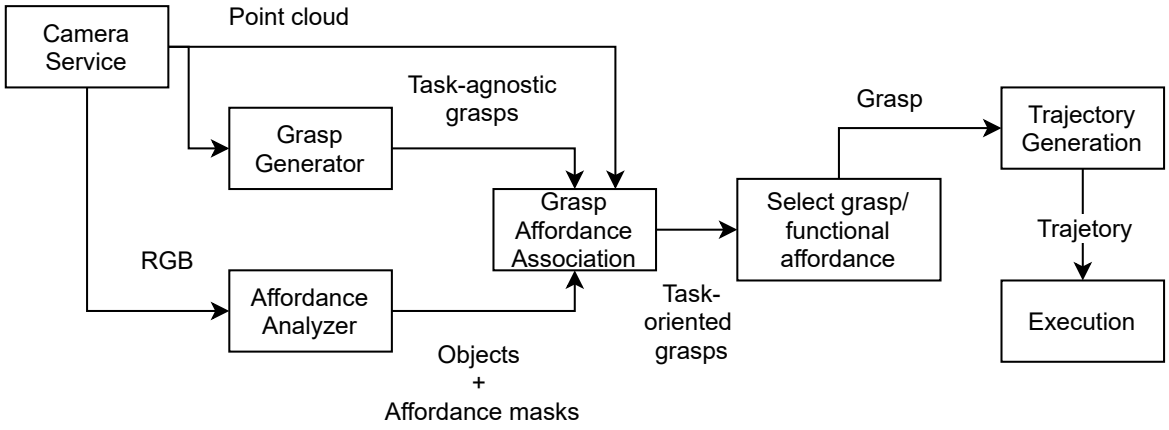


Figure 4.4: All of the modules involved in the task-oriented grasping test with a visualization of how information flows between the modules.

Metrics

The task-oriented grasping capabilities of the system are evaluated with the following metrics:

- Success: Binary, the robot is able to grasp the intended affordance of the desired object and transport it to a predefined pose without encountering any failures.
- Failure to generate grasps: Binary, the *grasp synthesis* module could not produce any grasps for the relevant affordance or failed to generate trajectory for any of the suggested grasps. Failure to generate trajectory is assumed to be failure to generate grasps because the *grasp synthesis* module failed to produce feasible grasps.
- Failure to grasp: Binary, the robot executed a grasp but failed to grasp and stabilise the object, thus not grasping the object.
- Failure to grasp and hold: Binary, the robot succeeded in grasping the object, but dropped it during transportation to a predefined pose.
- Failure to grasp correct affordance: The robot successfully grasped the object, but did not grasp the intended affordance.

Procedure

The task-oriented grasp test was conducted on a subset of the object set presented in Section 2.5, more specifically a hammer, a knife and a spatula. This subset was selected as each object has a *grasp* affordance and a *functional* affordance e.g., hammer has a *grasp* and *pound* affordance.

For each object, grasps for the handover and grasp task were attempted. For the handover task, the *grasp* affordance should be left unobstructed for the receiver to grasp. Thus, the system was told to grasp the *functional* affordance present in the affordance segmentation. For the grasp task, the intention is to grasp the *grasp* affordance and not any other affordances.

The following is the procedure for the task-oriented grasping test.

1. Place an object from the object set in front of the robot such that the object is fully in the field of view of the camera.
2. Confirm that the *affordance analysis* module is capable of detecting and segmenting the object. If not, return to step 1.
3. Execute the program
4. Repeat 5 times for each of the given tasks i.e., handover or grasp.
5. Repeat for each object in the object set.

Following the test procedure, 15 tests are performed for each task of handover and grasp for a total of 30 tests. The test is programmed and available as a script on the project's GitHub repository [93]. The results of the task-oriented grasping test can be seen in Section 5.3.

4.4 Full system test

This section describes the test of the full pipeline for the task-oriented handover with all modules working together.

Information flow

The information flow for the test is the system overview itself that can be seen in Figure 3.1. In this test the output of the *grasp-affordance association* module is only the grasps not associated with the *grasp* and *wide-grasp* affordances, therefore, the robot will grasp the *functional* affordance. Additionally, the handover will only be executed after an object to grasp has been identified with the *speech recognition* module.

Metrics

The full pipeline for doing task-oriented handover is evaluated with the following metrics:

- Success: Binary, the robot is able to grasp the requested affordance of the desired object and transport it to a predefined pose where it is handed over to the receiver without any failures.

- Failure to initiate: Binary, failure to recognize speech and initiate handover execution. Because running the robot creates a humming background noise, the user is allowed to repeat the request until the module registers the input. The failure only occurs if the user's command was deciphered incorrectly. For example, the user asks for a bowl, but the module understood ball.
- Failure to detect: Binary, failure to detect objects correctly not allowing handover to proceed.
- Failure to segment: Binary, failure to segment objects correctly not allowing handover to proceed.
- Failure to generate grasps: Binary, the *grasp synthesis* module could not produce any grasps for the relevant affordance or failed to generate trajectory for any of the suggested grasps. Failure to generate trajectory is assumed to be failure to generate grasps because the *grasp synthesis* module failed to produce feasible grasps.
- Failure to grasp: Binary, the robot executed a grasp but failed to grasp and stabilise the object, thus not grasping the object.
- Failure to grasp and hold: Binary, the robot succeeded in grasping the object, but dropped it during transportation to a predefined pose.
- Failure to grasp correct affordance: The robot successfully grasped the object, but did not grasp the intended affordance.
- Failure to handover: Binary, any failure after the object is moved to the handover location, resulting in the receiver not obtaining the tool.

Procedure

The task-oriented handover test was made with the subset of the object set presented in Section 2.5. This subset consisted of a hammer, a knife, and a spatula. This subset was selected as each object has boht a *grasp* affordance and a *functional* affordance. The three authors will be the test participants and act as the receiver during handover. This test seeks to evaluate handover and as such, the targeted affordances for the robot to grasp are any not associated with the *grasp* affordance. This allows the *grasp* affordance part of the object, in this case the handles of the tools, to be presented to the receiver.

The following is the procedure for the task-oriented handover test.

1. Place an object from the object set in front of the robot such that the object is fully in the field of view of the camera.
2. Start the program.
3. Wait for the system to compute all available task-oriented grasps.

4. Receiver asks for a specific object by naming the object.
5. Robot grasps and moves the desired object to a fixed position where the object is handed to the receiver.
6. Repeat 10 times for each object.
7. Repeat for each of the 3 test participant.

Following the procedure 30 test are performed for each object, resulting in a total of 90 performed tests of the full system. The results of the full system test can be seen in Section 5.4. The link to a demonstration video[1] can be found in Appendix B.

[1] Video: `https://drive.google.com/file/d/1JktlaXDOhjj8N5gfHmTbU9nxQGVFOxy3/view?usp=sharing`

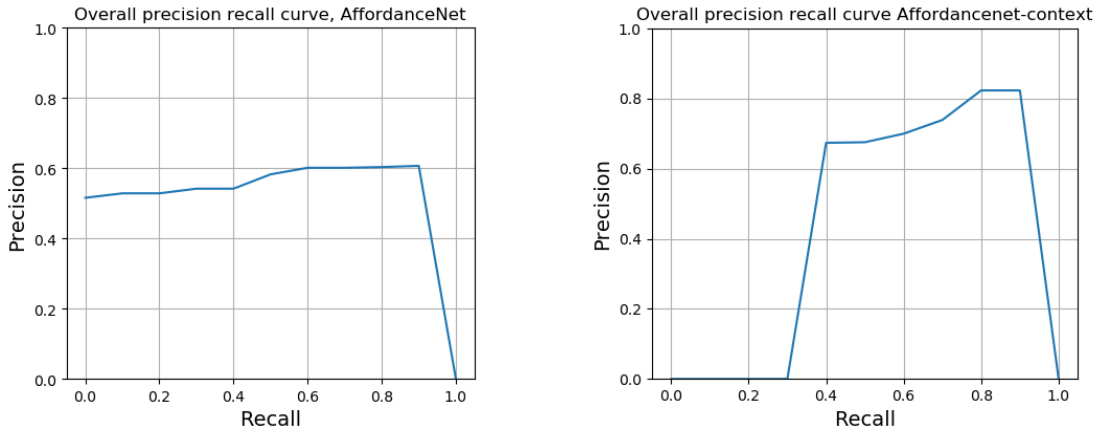
5 - Results

5.1 Affordance object detection

The results of the affordance object detection test can be see in Table 5.1. The mAP of AffordanceNet was computed from the overall precission recall seen in Figure 5.1a. The mAP of AffordanceNet-context was computed from the overall precission recall seen in Figure 5.1b.

	AffordanceNet	AffordanceNet-context
mAP	**0.604**	**0.495**

Table 5.1: Results of affordance object detection test.



(a) **(b)**

Figure 5.1: (a) shows the overall precision-recall curve of AffordanceNet. **(b)** shows the overall precision-recall curve of AffordanceNet-context.

5.2 Task-agnostic grasping test

Table 5.2 shows the result of the task-agnostic grasping test, which was performed as described in Section 4.2.

Object	Hammer	Bowl	Spatula	Bottle	Cup	Knife	**Average**
Success	100 %	100 %	100 %	100 %	100 %	80 %	**96.67 %**
Failed to generate grasps	0 %	0 %	0 %	0 %	0 %	0 %	**0 %**
Failed to grasp	0 %	0 %	0 %	0 %	0 %	20 %	**3.33 %**
Failed to grasp and hold	0 %	0 %	0 %	0 %	0 %	0 %	**0 %**

Table 5.2: Result of the task-agnostic grasping test.

5.3 Task-oriented grasping test

Table 5.3 and Table 5.4 show the results of the task-oriented grasping test, which was performed as described in Section 4.3.

Object	Knife		Spatula		Hammer	
Affordance	Grasp	Functional	Grasp	Functional	Grasp	Functional
Success	80.00 %	100.00 %	60.00 %	100.00 %	100.00 %	40.00 %
Failed to generate grasps	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
Failed to grasp	0.00 %	0.00 %	40.00 %	0.00 %	0.00 %	60.00 %
Failed to grasp and hold	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
Failed to grasp correct affordance	20.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %

Table 5.3: Result of the task-oriented grasping test. The *grasp* affordance refers to the grasp task, and the *functional* affordance refers to the handover task.

	Average	
Affordance	**Grasp**	**Functional**
Success	**80.00 %**	**80.00 %**
Failed to generate grasps	**0.00 %**	**0.00 %**
Failed to grasp	**13.33 %**	**20.00 %**
Failed to grasp and hold	**0.00 %**	**0.00 %**
Failed to grasp correct affordance	**6.67 %**	**0.00 %**

Table 5.4: Averaged results of the task-oriented grasping test. The *grasp* affordance refers to the grasp task, and the *functional* affordance refers to the handover task.

5.4 Full system test

Table 5.5 shows the result of the full system test, which was performed as described in Section 4.4. It should be noted here, that the test was performed with the lights off, which in turn increased the success rate of the object detection significantly. This implies that the pre-trained weights used for AffordanceNet were trained on a dataset that is not diverse enough illumination-wise.

Object	Knife	Spatula	Hammer	**Average**
Success	86.67 %	76.67 %	53.33 %	**72.22 %**
Failed to initiate	3.33 %	0.00 %	0.00 %	**1.11 %**
Failed to detect	0.00 %	0.00 %	0.00 %	**0.00 %**
Failed to segment	0.00 %	3.33 %	0.00 %	**1.11 %**
Failed to generate grasps	0.00 %	0.00 %	13.33 %	**4.44 %**
Failed to grasp	3.33 %	16.67 %	30.00 %	**16.67 %**
Failed to grasp and hold	0.00 %	3.33 %	0.00 %	**1.11 %**
Failed to grasp correct affordance	6.67 %	0.00 %	3.33 %	**3.33 %**
Failed to handover	0.00 %	0.00 %	0.00 %	**0.00 %**

Table 5.5: Result of the full system test.

6 - Discussion

This project set out to propose a novel pipeline for the problem of task-oriented handover. How to handover different objects can be especially challenging to solve. The semantic understanding of the objects that allow humans to understand what part of the object is appropriate to grasp in the handover situation is not natural to the robots. However, the affordance theory can help to address this issue.

6.1 Testing results

The modules of the system have been tested both individually and as part of the full pipeline. Requirements for the tests performed are presented in Section 2.6. Results are shown in Section 5.

6.1.1 Affordance object detection

The affordance object detection test identified a part of the system where performance is not as good as expected from a SOTA solution. AffordanceNet achieved a mAP of 0.604, which is low compared to SOTA object detectors [90]. However, it is worth noting that the performance seen from the other tested *affordance analysis* network was even lower, with the success rate at 0.495 mAP. The detection performance requirement of the *affordance* module is therefore not fulfilled.

6.1.2 Task-agnostic grasping test

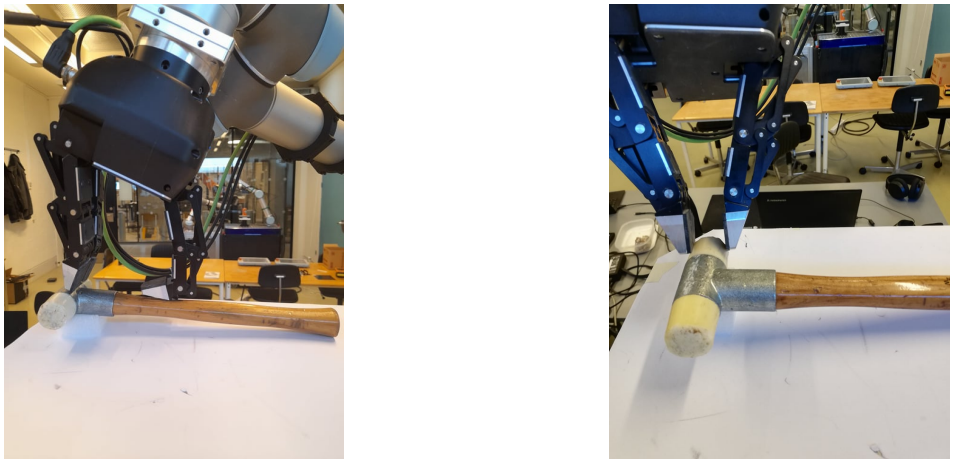
The result of the pure grasping test shows 96.67 % success rate. This fulfills the requirement of 80 % success rate. The only failure type during this test was failure to grasp. The success rate indicates that this module performs very well as a standalone module and should be a good addition to the full pipeline.

6.1.3 Task-oriented grasping test

The task-oriented grasping test showed generally good results with 80 % success rate for grasping both the *grasp* and *functional* affordance. Therefore, the success rate of grasping the target affordance exceeds the requirement of 70 %.

Two types of errors were recorded during the testing - failure to grasp and failure to grasp correct affordance. A failure to grasp correct affordance occurred only once and it was caused by a faulty affordance segmentation. A failure to grasp was more frequent cause of the issues.

There were two different types of grasp failures observed during testing. One, where the grasp generated looks feasible, but the gripper did not grip securely enough, resulting in the object not being lifted. This could be due to the gripper not gripping with high enough force or not gripping a large enough part of the object. These failures was especially seen when trying to grasp the head of the hammer, shown in Figure 6.1b. The second grasp failure was when an infeasible grasp was attempted, shown in Figure 6.1a. The cause of this is either that no good grasps were generated, good grasps were filtered out when selecting a desired affordance, or that the *trajectory generation* module found a valid trajectory for a low scoring grasp. This failure could be alleviated by setting a cut-off threshold for grasp scores. If no grasp above this threshold is feasible, a new set of grasp should be generated. Further experimentation would be required to determine a good score cut-off, but results from the authors of GraspNet shown in Table 3.1 suggests that a score cut-off somewhere between 0.5 and 1 would be appropriate. The fact that the gripper used the Robotiq 3-finger gripper might also result in some grasp failures that could have succeeded using a parallel gripper.



(a) **(b)**

Figure 6.1: Different grasp failures observed during testing. **(a)** A grasp that is infeasible with no way of grasping the hammer **(b)** A grasp failure where the gripper does not get a secure grasp on the object resulting in a drop while lifting.

6.1.4 Speech test

The results of *speech recognition* in the standalone testing showed a 92 % success rate. This showed that the module was capable of detecting keywords in speech which allows it to extract a desired object and affordance to pick up. This result was deemed sufficient for using it in the full solution, where it served as the initiation for the handover. It was responsible for 1.11 % of the failures of the full system. All failures of this module were due to speech-to-text not returning the correct text. This resulted in no keyword being found in the text and, thereby, no tool ID being sent. In practice, the solution to this would be to ask the robot for the object again.

6.1.5 Full system test

A total of 90 test runs was performed for the full system test. The full system test concluded with a success rate of 72.22 %, which exceeds the requirement of 65 %. During the test, four different types of failures where experienced - failure to grasp 16.67 %, failure to generate grasps 4.44 %, failure to grasp correct affordance 3.33 %, failure to segment 1.11 %, and failure to initiate 1.11 %. Failure to grasp being the most prominent type of failure aligns with the results acquired from the grasping and task-oriented grasping test. The underlying issues remains the same for all three tests. From all the generated grasps that are associated with the user's requested tool, the first feasible grasp is selected and executed. However, there is no guarantee that the first feasible grasp is also a high-scoring one.

Failure to generate grasp occurs when either no grasp are generated or MoveIt fails to generate a trajectory to any of the generated grasps. No cases during testing showed no generated grasps. Failing to generate a trajectory for the grasps could be resolved with running multiple trajectory planning attempts. RRTConnect, the planing algorithm of the proposed pipeline, explores the state space randomly, which means that two separate instances with the same start and goal state may result in two different solutions. On the other hand, re-running planning increases planning time and delays the handover execution. Furthermore, there exist no guarantee that the generated grasps are reachable by the robot.

Failure to segment and failure to grasp correct affordance are tightly coupled with the performance of AffordanceNet. As was stated in Section 2.3, AffordanceNet achieved average F_β^w of 0.799 on the UMD dataset and 0.734 on IIT-AFF. While both scores represent state-of-the-art performance in the field of affordance segmentaion, there is still margin for an error. Similarly to the trajectory segmentation, the failure to segment could be resolved with running AffordanceNet until the desired segmentation is found. However, AffordanceNet, as implemented in the pipeline, is the slowest performing module and re-running it would significantly increase the handover time.

The result of the final test shows that the developed system is able to achieve results with similar success rate to other work in task-oriented grasping, such as Detry et al. [37]

which grasped different affordances with 69 % success rate. Chu et al. [42] also performs task-oriented grasping and object manipulation with a 82.5 % success rate. While similar in that the grasp considers affordances neither of these pieces of previous work performs handovers as part of the solution.

While the success rate of 72.22 % leaves room for improvement, to the best of the authors' knowledge, it is the first recorded full robot-to-human task-oriented handover system in recent years. As shown in this report, research has been made into sub-problems of the task-oriented handover e.g., the task-oriented grasping, but not much information can be found on the topic of the task-oriented handover itself. Therefore, the results achieved in the full system test can serve as a base for later solutions proposing to solve the task-oriented handover using explicit affordance knowledge and SOTA task-agnostic grasping.

6.2 Novelty

The system utilises grasp-affordance association as a novel way to generate a few simple rules that apply to a wide variety of objects. This is especially interesting for even bigger object sets than used in this project. The case could easily be that a set of object classes is several orders of magnitude larger than the set of affordances that describes their functions. In such a case, rules about only the affordances could be made to control the grasping behaviour of the system. An example of such a rule in this project was leaving the *grasp* and *wide-grasp* affordance available, so the receiver would be easily able to grasp the object. This resulted in a general rule that could be applied to all the tools used during testing, where the grasps were generated for the other *functional* affordances, leaving the handle available for the receiver. As the system is in its current iteration, the *affordance analysis* module is the limiting factor for the number of object classes that can be worked with. The *grasp synthesis* module is capable of generating grasps for novel objects. The object detection of the *affordance analysis* module could be changed to either detect more object classes or use a general object detector as the AffordanceNet-Context network proposed by Chu et al. does [83]. This would allow the system to generalise to many different situations only requiring changes to the affordance rules.

6.3 System

The modular approach of the system allows for an easy change of individual modules if a new SOTA method becomes available. As long as the input/output of the module remains the same, it would be able to be exchanged without needing to change other module functionality. The novel method developed for grasp-affordance association would work with any 6-DOF *grasp synthesis* module and *affordance analysis* module that outputs affordance masks.

The system presented in this work serves as a foundation for further development. Even better results could be achieved using this methodology with further tuning of grasping alongside improvements to affordance analysis. Potential areas of future work with this system is presented in Section 6.4.

6.4 Future work

This section covers opportunities for future work based on the aspects of the solution that were identified for further development during the implementation and testing.

- The affordance analysis did not perform well in terms of processing time and object detection success rate. This module could use further development by training AffordanceNet on a new and larger dataset to achieve better object detection and help the illumination issues observed. Alternatively, developing a new affordance network could be an option.
- The *grasp synthesis* module is based on GraspNet [58]. A new grasping network from the same authors achieves a higher performance on the GraspNet-1Billion dataset [98]. When an implementation of this new network is made available, it could be implemented as an upgrade to the *grasp synthesis* module.
- The *speech recognition* could be further developed to learn the object-affordance relationships. This will enable the *speech recognition* to give suggestions or disallow invalid combinations, like asking for the *cut* affordance of a cup.
- Even though it was not the focus of this project, the addition of human detection to the handover pipeline could come with several benefits, such as not having a fixed handover location but instead having it fit the human location and being able to consider the ergonomics of the handover based on attributes like the height of the person.
- Testing how the task-oriented handover from the current solution impacts the experience of the receiver. This should be done by comparing two tests. A test with and a test without considering affordance when grasping. Requires test participants outside the group of authors.
- Testing the robustness of the pipeline and developed approach with different hardware, such as grippers and robots.
- The current pipeline is developed with open-loop grasping. Closed-loop grasping would make the robot able to update grasp as it approaches the object and adjust for object movement during execution. With a camera mounted on the gripper, the point cloud could be updated while approaching the object and allow for better *grasp synthesis*.

- Generating task-oriented trajectories is a topic that could be further investigated, especially in relation to affordances. A cup containing a liquid such as water has an additional affordance called *pour* compared to an empty cup. The *pour* affordance impose extra constraints on trajectory as the cup contraining a liquid should be ideally transported without spilling. As such, an investigation into the constraints imposed by the object affordances should be investigated.

6.5 Conclusion

This report presents a novel solution that combines a task-agnostic grasping SOTA method GraspNet with an affordance segmentation SOTA neural netwok AffordanceNet in order to successfully perform task-oriented handovers. The proposed solution is speech initiated but currently does not modify the handover location and orientation based on the user's pose. As the main contribution of this report, the authors identify the *grasp-affordance association* module that automatically groups task-agnostic grasps with a user-desired affordance. The solution was developed within the ROS framework. The authors believe that the proposed pipeline shows how can how a UR5 cobot equipped with an Intel RealSense D435 and a Robotiq 3-finger gripper can perform successful task-oriented handovers of objects from a custom object set by combining task-agnostic grasping methods with affordance segmentation

During the final system test, the solution achieved the success rate of 72.22 %. While the authors acknowledge that this success rate is not high-enough for real world applications, the recorded success rate is, to the best of the authors' knowledge, the first of its kind as the data on the task-oriented handover is scarce. Therefore, the above-mentioned success rate can serve as a baseline for future research into the problem.

Bibliography

[1] Patrick Rosenberger, Akansel Cosgun, Rhys Newbury, Jun Kwan, Valerio Ortenzi, Peter Corke, and Manfred Grafinger. "Object-independent human-to-robot handovers using real time robotic vision". In: *IEEE Robotics and Automation Letters* 6.1 (2020), pp. 17–23.

[2] Wei Yang, Chris Paxton, Maya Cakmak, and Dieter Fox. *Human Grasp Classification for Reactive Human-to-Robot Handovers*. 2020. arXiv: 2003.06000 [cs.RO].

[3] Paola Ardón, Maria E. Cabrera, Èric Pairet, Ronald P. A. Petrick, Subramanian Ramamoorthy, Katrin S. Lohan, and Maya Cakmak. *Affordance-Aware Handovers with Human Arm Mobility Constraints*. 2021. arXiv: 2010.15436 [cs.RO].

[4] Valerio Ortenzi, Akansel Cosgun, Tommaso Pardi, Wesley Chan, Elizabeth Croft, and Dana Kulic. *Object Handovers: a Review for Robotics*. 2020. arXiv: 2007.12952 [cs.RO].

[5] Andrea Mason and Christine MacKenzie. "Grip forces when passing an object to a partner". In: *Experimental brain research. Experimentelle Hirnforschung. Expérimentation cérébrale* 163 (June 2005), pp. 173–87. DOI: 10.1007/s00221-004-2157-x.

[6] Natalie Sebanz, Harold Bekkering, and Günther Knoblich. "Joint action: bodies and minds moving together". In: *Trends in Cognitive Sciences* 10.2 (2006), pp. 70–76. ISSN: 1364-6613. DOI: https://doi.org/10.1016/j.tics.2005.12.009. URL: https://www.sciencedirect.com/science/article/pii/S1364661305003566.

[7] Valerio Ortenzi, Francesca Cini, Tommaso Pardi, Naresh Marturi, Rustam Stolkin, Peter Corke, and Marco Controzzi. "The grasp strategy of a robot passer influences performance and quality of the robot-human object handover". In: *Frontiers in Robotics and AI* 7 (2020).

[8] SH Creem and DR Proffitt. "Grasping objects by their handles: a necessary interaction between cognition and action". In: *Journal of experimental psychology. Human perception and performance* 27.1 (2001), 218—228. ISSN: 0096-1523. DOI: 10.1037//0096-1523.27.1.218. URL: https://doi.org/10.1037//0096-1523.27.1.218.

[9] J.J. Gibson. "The Theory of Affordances". In: (1977).

[10] Yixin Zhu, Yibiao Zhao, and Song-Chun Zhu. "Understanding tools: Task-oriented object modeling, learning and recognition". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 2855–2864. DOI: 10.1109/CVPR.2015.7298903.

[11] Hema S. Koppula and Ashutosh Saxena. "Anticipating Human Activities Using Object Affordances for Reactive Robotic Response". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.1 (2016), pp. 14–29. DOI: 10.1109/TPAMI.2015.2430335.

[12] Thanh-Toan Do, Anh Nguyen, and Ian Reid. *AffordanceNet: An End-to-End Deep Learning Approach for Object Affordance Detection*. 2018. arXiv: 1709.07326 [cs.CV].

[13] Mads Hvilshøj and Simon Bøgh. ""Little Helper" — An Autonomous Industrial Mobile Manipulator Concept". In: *International Journal of Advanced Robotic Systems* 8.2 (2011), p. 15. DOI: 10.5772/10579. eprint: https://doi.org/10.5772/10579. URL: https://doi.org/10.5772/10579.

[14] Jens F. Buhl, Rune Grønhøj, Jan K. Jørgensen, Guilherme Mateus, Daniela Pinto, Jacob K. Sørensen, Simon Bøgh, and Dimitrios Chrysostomou. "A Dual-arm Collaborative Robot System for the Smart Factories of the Future". In: *Procedia Manufacturing* 38 (2019). 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing, pp. 333–340. ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2020.01.043. URL: https://www.sciencedirect.com/science/article/pii/S2351978920300445.

[15] Ansgar Koene, Anthony Remazeilles, Miguel Prada, Ainara Garzo, Mildred Puerto, Satoshi Endo, and Alan M Wing. "Relative importance of spatial and temporal precision for user satisfaction in human-robot object handover interactions". In: *Third International Symposium on New Frontiers in Human-Robot Interaction*. 2014.

[16] Alap Kshirsagar, Hadas Kress-Gazit, and Guy Hoffman. "Specifying and synthesizing human-robot handovers". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 5930–5936.

[17] Marco Controzzi, Harmeet Singh, Francesca Cini, Torquato Cecchini, Alan Wing, and Christian Cipriani. "Humans adjust their grip force when passing an object according to the observed speed of the partner's reaching out movement". In: *Experimental brain research* 236.12 (2018), pp. 3363–3377.

[18] Solly Brown and Claude Sammut. "Tool Use Learning in Robots". In: *AAAI Fall Symposium: Advances in Cognitive Systems*. 2011.

[19] Maya Cakmak, Siddhartha S Srinivasa, Min Kyung Lee, Jodi Forlizzi, and Sara Kiesler. "Human preferences for robot-human hand-over configurations". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 1986–1993.

[20] Markus Huber, Markus Rickert, Alois Knoll, Thomas Brandt, and Stefan Glasauer. "Human-robot interaction in handing-over tasks". In: *RO-MAN 2008-The 17th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE. 2008, pp. 107–112.

[21] Robin Rasch, Sven Wachsmuth, and Matthias König. "Understanding movements of hand-over between two persons to improve humanoid robot systems". In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE. 2017, pp. 856–861.

[22] Robin Rasch, Sven Wachsmuth, and Matthias König. "A joint motion model for human-like robot-human handover". In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2018, pp. 180–187.

[23] Robin Rasch, Sven Wachsmuth, and Matthias König. "Combining Cartesian Trajectories with Joint Constraints for Human-Like Robot-Human Handover". In: *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2019, pp. 91–98.

[24] Sina Parastegari, Bahareh Abbasi, Ehsan Noohi, and Miloš Zefran. "Modeling human reaching phase in human-human object handover with application in robot-human handover". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 3597–3602.

[25] Jacopo Aleotti, Vincenzo Micelli, and Stefano Caselli. "An affordance sensitive system for robot to human object handover". In: *International Journal of Social Robotics* 6.4 (2014), pp. 653–666.

[26] Wesley P Chan, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. "Determining proper grasp configurations for handovers through observation of object movement patterns and inter-object interactions during usage". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 1355–1360.

[27] Wesley P Chan, Matthew KXJ Pan, Elizabeth A Croft, and Masayuki Inaba. "Characterization of handover orientations used by humans for efficient robot to human handovers". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 1–6.

[28] Wesley P Chan, Matthew KXJ Pan, Elizabeth A Croft, and Masayuki Inaba. "An affordance and distance minimization based method for computing object orientations for robot human handovers". In: *International Journal of Social Robotics* 12.1 (2020), pp. 143–162.

[29] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. *Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics*. 2017. arXiv: 1703.09312 [cs.RO].

[30] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. *6-DOF GraspNet: Variational Grasp Generation for Object Manipulation*. 2019. arXiv: 1905.10520 [cs.CV].

[31] Douglas Morrison, Peter Corke, and Jürgen Leitner. "Learning robust, real-time, reactive robotic grasping". In: *The International Journal of Robotics Research* 39.2-3 (2020), pp. 183–201. DOI: 10.1177/0278364919859066. eprint: https://doi.org/10.1177/0278364919859066. URL: https://doi.org/10.1177/0278364919859066.

[32] Z. Li and S.S. Sastry. "Task-oriented optimal grasping by multifingered robot hands". In: *IEEE Journal on Robotics and Automation* 4.1 (1988), pp. 32–44. DOI: 10.1109/56.769.

[33] R. Haschke, J.J. Steil, I. Steuwer, and H. Ritter. "Task-oriented quality measures for dextrous grasping". In: *2005 International Symposium on Computational Intelligence in Robotics and Automation*. 2005, pp. 689–694. DOI: 10.1109/CIRA.2005.1554357.

[34] Mario Prats, Pedro J. Sanz, and Angel P. del Pobil. "Task-Oriented Grasping using Hand Preshapes and Task Frames". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, pp. 1794–1799. DOI: 10.1109/ROBOT.2007.363582.

[35] D. Song, K. Huebner, V. Kyrki, and D. Kragic. "Learning task constraints for robot grasping using graphical models". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 1579–1585. DOI: 10.1109/IROS.2010.5649406.

[36] Hao Dang and Peter K. Allen. "Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 1311–1317. DOI: 10.1109/IROS.2012.6385563.

[37] Renaud Detry, Jeremie Papon, and Larry Matthies. "Task-oriented grasping with semantic and geometric scene understanding". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 3266–3273. DOI: 10.1109/IROS.2017.8206162.

[38] Luca Cavalli, Gianpaolo Di Pietro, and Matteo Matteucci. "Towards affordance prediction with vision via task oriented grasp quality metrics". In: *arXiv preprint arXiv:1907.04761* (2019).

[39] Anh Nguyen, Dimitrios Kanoulas, Darwin G. Caldwell, and Nikos G. Tsagarakis. "Object-based affordances detection with Convolutional Neural Networks and dense Conditional Random Fields". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 5908–5915. DOI: 10.1109/IROS.2017.8206484.

[40] Mia Kokic, Johannes A Stork, Joshua A Haustein, and Danica Kragic. "Affordance detection for task-specific grasping using deep learning". In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE. 2017, pp. 91–98.

[41] Paola Ardón, Éric Pairet, Ronald PA Petrick, Subramanian Ramamoorthy, and Katrin S Lohan. "Learning grasp affordance reasoning through semantic relations". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4571–4578.

[42] Fu-Jen Chu, Ruinian Xu, Landan Seguin, and Patricio A. Vela. "Toward Affordance Detection and Ranking on Novel Objects for Real-World Robotic Manipulation". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4070–4077. DOI: 10.1109/LRA.2019.2930364.

[43] Fu-Jen Chu, Ruinian Xu, and Patricio A. Vela. "Learning Affordance Segmentation for Real-World Robotic Manipulation via Synthetic Images". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1140–1147. DOI: 10.1109/LRA.2019.2894439.

[44] Weiyu Liu, Angel Daruna, and Sonia Chernova. "Cage: Context-aware grasping engine". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 2550–2556.

[45] Andreas Ten Pas and Robert Platt. "Using geometry to detect grasp poses in 3d point clouds". In: *Robotics Research*. Springer, 2018, pp. 307–324.

[46] Muhayy Ud Din, M Usman Sarwar, Imran Zahoor, Wajahat M Qazi, and Jan Rosell. "Learning action-oriented grasping for manipulation". In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE. 2019, pp. 1575–1578.

[47] Adithyavairavan Murali, Weiyu Liu, Kenneth Marino, Sonia Chernova, and Abhinav Gupta. "Same Object, Different Grasps: Data and Semantic Knowledge for Task-Oriented Grasping". In: *Conference on Robot Learning*. 2020.

[48] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. "Learning task-oriented grasping for tool manipulation from simulated self-supervision". In: *The International Journal of Robotics Research* 39.2-3 (2020), pp. 202–216. DOI: 10.1177/0278364919872545. eprint: https://doi.org/10.1177/0278364919872545. URL: https://doi.org/10.1177/0278364919872545.

[49] Mia Kokic, Danica Kragic, and Jeannette Bohg. "Learning Task-Oriented Grasping From Human Activity Datasets". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3352–3359. DOI: 10.1109/LRA.2020.2975706.

[50] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. "Data-Driven Grasp Synthesis—A Survey". In: *IEEE Transactions on Robotics* 30.2 (2014), pp. 289–309. DOI: 10.1109/TRO.2013.2289018.

[51] I-Ming Chen and J.W. Burdick. "Finding antipodal point grasps on irregularly shaped objects". In: *IEEE Transactions on Robotics and Automation* 9.4 (1993), pp. 507–512. DOI: 10.1109/70.246063.

[52] Antonio Bicchi. "On the closure properties of robotic grasping". In: *The International Journal of Robotics Research* 14.4 (1995), pp. 319–334.

[53] Minghao Gou, Hao-Shu Fang, Zhanda Zhu, Sheng Xu, Chenxi Wang, and Cewu Lu. *RGB Matters: Learning 7-DoF Grasp Poses on Monocular RGBD Images*. 2021. arXiv: 2103.02184 [cs.RO].

[54] Yun Jiang, Stephen Moseson, and Ashutosh 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.

[55] Hongzhuo Liang, Xiaojian Ma, Shuang Li, Michael Gorner, Song Tang, Bin Fang, Fuchun Sun, and Jianwei Zhang. "PointNetGPD: Detecting Grasp Configurations from Point Sets". In: *2019 International Conference on Robotics and Automation (ICRA)* (2019). DOI: 10.1109/icra.2019.8794435. URL: http://dx.doi.org/10.1109/ICRA.2019.8794435.

[56] Joseph Redmon and Anelia Angelova. *Real-Time Grasp Detection Using Convolutional Neural Networks*. 2015. arXiv: 1412.3128 [cs.RO].

[57] Umar Asif, Mohammed Bennamoun, and Ferdous A. Sohel. "RGB-D Object Recognition and Grasp Detection Using Hierarchical Cascaded Forests". In: *IEEE Transactions on Robotics* 33.3 (2017), pp. 547–564. DOI: 10.1109/TRO.2016.2638453.

[58] Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. "GraspNet-1Billion: A Large-Scale Benchmark for General Object Grasping". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*. 2020, pp. 11444–11453.

[59] Douglas Morrison, Peter Corke, and Jürgen Leitner. *Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach*. 2018. arXiv: 1804.05172 [cs.RO].

[60] Chiara Gabellieri, Franco Angelini, Visar Arapi, Alessandro Palleschi, Manuel G. Catalano, Giorgio Grioli, Lucia Pallottino, Antonio Bicchi, Matteo Bianchi, and Manolo Garabini. "Grasp It Like a Pro: Grasp of Unknown Objects With Robotic Hands Based on Skilled Human Expertise". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2808–2815. DOI: 10.1109/LRA.2020.2974391.

[61] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. "Benchmarking in Manipulation Research: Using the Yale-CMU-Berkeley Object and Model Set". In: *IEEE Robotics & Automation Magazine* 22.3 (2015), 36–52. ISSN: 1070-9932. DOI: 10.1109/mra.2015.2448951. URL: http://dx.doi.org/10.1109/MRA.2015.2448951.

[62] Jürgen Leitner, Adam W. Tow, Niko Sünderhauf, Jake E. Dean, Joseph W. Durham, Matthew Cooper, Markus Eich, Christopher Lehnert, Ruben Mangels, Christopher McCool, Peter T. Kujala, Lachlan Nicholson, Trung Pham, James Sergeant, Liao Wu, Fangyi Zhang, Ben Upcroft, and Peter Corke. "The ACRV picking benchmark: A robotic shelf picking benchmark to foster reproducible research". In: *2017 IEEE In-*

ternational Conference on Robotics and Automation (ICRA). 2017, pp. 4705–4712. DOI: 10.1109/ICRA.2017.7989545.

[63] Antonio Morales, Mario Prats, and Javier Felip. "Sensors and Methods for the Evaluation of Grasping". In: *Grasping in Robotics*. Ed. by Giuseppe Carbone. London: Springer London, 2013, pp. 77–104. ISBN: 978-1-4471-4664-3. DOI: 10.1007/978-1-4471-4664-3_4. URL: https://doi.org/10.1007/978-1-4471-4664-3_4.

[64] Kilian Kleeberger, Richard Bormann, W. Kraus, and M. Huber. "A Survey on Learning-Based Robotic Grasping". In: 2020.

[65] Jia-Wei Li, Hong Liu, and He-Gao Cai. "On computing three-finger force-closure grasps of 2-D and 3-D objects". In: *IEEE Transactions on Robotics and Automation* 19.1 (2003), pp. 155–161. DOI: 10.1109/TRA.2002.806774.

[66] A.T. Miller, S. Knoop, H.I. Christensen, and P.K. Allen. "Automatic grasp planning using shape primitives". In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 2. 2003, 1824–1829 vol.2. DOI: 10.1109/ROBOT.2003.1241860.

[67] A. Sahbani, S. El-Khoury, and P. Bidaud. "An overview of 3D object grasp synthesis algorithms". In: *Robotics and Autonomous Systems* 60.3 (2012). Autonomous Grasping, pp. 326–336. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2011.07.016. URL: https://www.sciencedirect.com/science/article/pii/S0921889011001485.

[68] Zhikai Dong, Sicheng Liu, Tao Zhou, Hui Cheng, Long Zeng, Xingyao Yu, and Houde Liu. "PPR-Net:Point-wise Pose Regression Network for Instance Segmentation and 6D Pose Estimation in Bin-picking Scenarios". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 1773–1780. DOI: 10.1109/IROS40897.2019.8967895.

[69] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. *Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects*. 2018. arXiv: 1809.10790 [cs.RO].

[70] Felix Spenrath and Andreas Pott. "Gripping Point Determination for Bin Picking Using Heuristic Search". In: *Procedia CIRP* 62 (Dec. 2017), pp. 606–611. DOI: 10.1016/j.procir.2016.06.015.

[71] Bernhard Schölkopf, John Platt, and Thomas Hofmann. "Robotic Grasping of Novel Objects". In: *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*. 2007, pp. 1209–1216.

[72] Shehan Caldera, Alexander Rassau, and Douglas Chai. "Review of Deep Learning Methods in Robotic Grasp Detection". In: *Multimodal Technologies and Interaction* 2.3 (2018). ISSN: 2414-4088. DOI: 10.3390/mti2030057. URL: https://www.mdpi.com/2414-4088/2/3/57.

[73] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2017. arXiv: 1612.00593 [cs.CV].

[74] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. 2017. arXiv: 1706.02413 [cs.CV].

[75] Mohammed Hassanin, Salman Khan, and Murat Tahtali. *Visual Affordance and Function Understanding: A Survey*. 2018. arXiv: 1807.06775 [cs.CV].

[76] Austin Myers, Ching L. Teo, Cornelia Fermüller, and Yiannis Aloimonos. "Affordance detection of tool parts from geometric features". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 1374–1381. DOI: 10.1109/ICRA.2015.7139369.

[77] Johann Sawatzky, Abhilash Srikantha, and Juergen Gall. "Weakly Supervised Affordance Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5197–5206. DOI: 10.1109/CVPR.2017.552.

[78] Shengheng Deng, Xun Xu, Chaozheng Wu, Ke Chen, and Kui Jia. *3D AffordanceNet: A Benchmark for Visual Object Affordance Understanding*. 2021. arXiv: 2103.16397 [cs.CV].

[79] Ran Margolin, Lihi Zelnik-Manor, and Ayellet Tal. "How to Evaluate Foreground Maps". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 248–255. DOI: 10.1109/CVPR.2014.39.

[80] Anh Nguyen, Dimitrios Kanoulas, Darwin G. Caldwell, and Nikos G. Tsagarakis. "Detecting object affordances with Convolutional Neural Networks". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2765–2770. DOI: 10.1109/IROS.2016.7759429.

[81] Ross Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV].

[82] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. *Mask R-CNN*. 2018. arXiv: 1703.06870 [cs.CV].

[83] Fu-Jen Chu, Ruinian Xu, Chao Tang, and Patricio A Vela. "Recognizing object affordances to support scene reasoning for manipulation tasks". In: *arXiv preprint arXiv:1909.05770* (2019).

[84] Chao Shi, Masahiro Shiomi, Christian Smith, Takayuki Kanda, and Hiroshi Ishiguro. "A Model of Distributional Handing Interaction for a Mobile Robot". In: *Robotics: science and systems*. 2013, pp. 24–28.

[85] Elena Corina Grigore, Kerstin Eder, Anthony G Pipe, Chris Melhuish, and Ute Leonards. "Joint action understanding improves robot-to-human object handover". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 4622–4629.

[86] Miguel Prada, Anthony Remazeilles, Ansgar Koene, and Satoshi Endo. "Implementation and experimental validation of dynamic movement primitives for object handover". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 2146–2153.

[87] Valerio Ortenzi, Marco Controzzi, Francesca Cini, Juxi Leitner, Matteo Bianchi, Maximo A Roa, and Peter Corke. "Robotic manipulation and the role of the task in the metric of success". In: *Nature Machine Intelligence* 1.8 (2019), pp. 340–346.

[88] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[89] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[90] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].

[91] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[92] GraspNet. *GraspNet-1Billion*. (visited on 20-11-2022). URL: \url{https://graspnet.net/datasets.html}.

[93] Albert D. Christensen, Daniel Lehotsky, and Marius W. Jørgensen. *Project's GitHub repository*. URL: \url{https://github.com/daniellehot/ROB9}.

[94] Anthony Zhang. *SpeechRecognition*. (visited on 12-11-2021). URL: https://github.com/Uberi/speech_recognition.

[95] Steven M LaValle et al. "Rapidly-exploring random trees: A new tool for path planning". In: (1998).

[96] James J Kuffner and Steven M LaValle. "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 2. IEEE. 2000, pp. 995–1001.

[97] Philipp Ruppel. "Performance optimization and implementation of evolutionary inverse kinematics in ROS". PhD thesis. Universität Hamburg, 2017.

[98] Chenxi Wang, Hao-Shu Fang, Minghao Gou, Hongjie Fang, Jin Gao, and Cewu Lu. "Graspness Discovery in Clutters for Fast and Accurate Grasp Detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15964–15973.

A - Questionnaire

1. Fluency
    - The human-robot team worked fluently together.
    - The human-robot team's fluency improved over time.
    - The robot contributed to the fluency of the interaction.
2. Trust
    - I trusted the robot to do the right thing at the right time.
    - The robot was trustworthy.
3. Working alliance
    - The robot accurately perceives what my goals are.
    - I understand what the robot's goals are.
    - The robot and I are working towards mutually agreed upon goals.

B - Demonstration video

The video demonstration can be found here:
https://drive.google.com/file/d/1JktlaXDOhjj8N5gfHmTbU9nxQGVFOxy3/view?usp=sharing