Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MASTERTHESIS

Bimanual Robot-to-Robot Handover Utilizing Multi-Modal Feedback

vorgelegt von

Björn Sygo

MIN-Fakultät

Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Studiengang: Master Informatik

Matrikelnummer: 6921896

Abgabedatum: 30.03.2024

Erstgutachter: Prof. Dr. Jianwei Zhang

Zweitgutachter: Dr. Norman Hendrich

Betreuer: Michael Görner

Abstract

Enabling bimanual robots to perform handovers from one manipulator to the other promises significant advantages when operating in complex environments, i.e., an increased workspace or easier object reorientation. However, implementing it in practice remains challenging. This thesis implemented a pipeline on a bimanual robot with an anthropomorphic hand. The pipeline can hand over different objects without previous knowledge of their shape. In this context, the thesis explores the available workspace of this system for such a task. Additionally, it investigates the possibility of using a reinforcement learning model trained exclusively on the real-world system to perform the grasping part of the pipeline. The training uses the concept of hand synergies to control the anthropomorphic hand during this process. Results indicate the viability of such an approach, where the model performed similarly to the baseline within a reasonable training time for real-world training.

Zusammenfassung

Zweiarmigen Robotern zu ermöglichen, sich Objekte von einem Manipulator zum anderen zu übergeben, verspricht erhebliche Vorteile beim Agieren in komplexen Umgebungen, zum Beispiel einen größeren Arbeitsbereich oder einfacheres Neuausrichten dieser Objekte. Die Umsetzung in der Praxis bleibt jedoch eine Herausforderung. Diese Arbeit implementiert solch eine Pipeline auf einem zweiarmigen Roboter mit einer anthropomorphen Hand. Diese ist in der Lage, verschiedene Objekte ohne vorherige Kenntnis ihrer Form zu übergeben. In diesem Zusammenhang untersucht die Arbeit weiterhin den verfügbaren Arbeitsbereich dieses Systems für eine solche Aufgabe. Darüber hinaus wird die Möglichkeit untersucht, ein ausschließlich auf dem realen System trainiertes Reinforcement-Learning-Modell zu verwenden, um in der Pipeline das Greifen der Objekte auszuführen. Das Training nutzt das Konzept der Handsynergien, um die anthropomorphe Hand während dieses Prozesses zu kontrollieren. Die Ergebnisse zeigen die Realisierbarkeit eines solchen Ansatzes, bei dem das Modell innerhalb einer für ein Training in der realen Welt angemessenen Trainingszeit eine ähnliche Leistung wie die Basislinie erbringt.

Contents

List of Figures

List of Tables

1. Introduction

With the constant rise of interest in robotics, more and more applications outside laboratories emerge. While robots in factories are already widespread, they often operate independently of humans, performing highly specialized tasks. However, newer developments aim to have robots operate beside humans and expand the deployment area to include other domains, such as domestic environments. Two requirements to succeed in this endeavor are the robot's capability to operate in these unstructured, human-centric environments and to stay predictable for humans in their movements. Consequently, a critical skill for robots is the ability to hand over objects. Humans constantly grasp objects and pass them to their other hand for further use. Examples include removing objects from a dishwasher or grasping a tool on the other side of their preferred hand while working.

Robots will encounter two general scenarios when performing handovers. The first involves a human and a robot, where the robot either gives or receives an object to or from the human. This field, therefore, encompasses human-to-robot and robot-to-human handovers [1]. However, this thesis focuses on the second type, involving two robot arms. These can either belong to two different agents or the same one. The latter case, where one robot performs a handover of an object from one of its manipulators to the other, is the scenario for this thesis. This scenario requires a robot platform equipped with two arms performing a bimanual task, as defined by Smith et al. [2].

Providing the capability of performing bimanual handovers on such a platform also comes with significant advantages. For example, it makes changing an object's orientation easier than through in-hand object manipulation. Enabling object re-grasping without needing a support surface also increases the flexibility of task assignments. Additionally, it allows them to move and use objects in a larger area without moving the rest of their bodies. As many service robots in development have a dual-arm setup, the last part is essential regarding predictable movements.

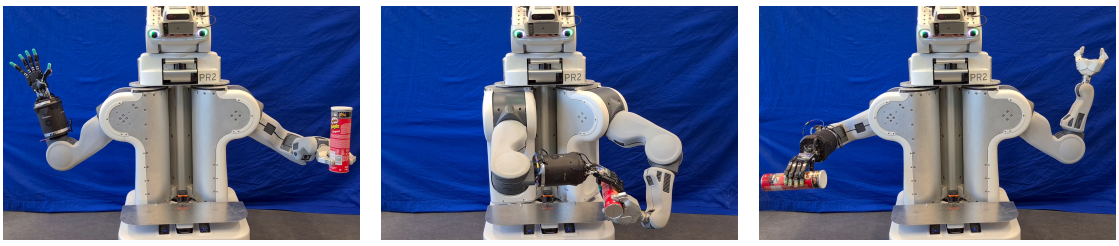


Figure 1.1.: An example of the robot used in this thesis handing an object from one manipulator to the other.

Therefore, the first aim of this thesis is to implement a pipeline capable of performing a bimanual handover on such a platform. It combines multiple modalities sensed by sensors on the robot to achieve this. The pipeline has to work with different objects. It also has no prior knowledge of the shape of these objects, as this would also not be the case when operating outside of a laboratory environment. To this extent, the thesis also performs a workspace analysis to implement this bimanual task properly [3].

However, as seen in Figure 1.1, the robot used in this thesis is also equipped with an anthropomorphic hand as one of its manipulators. Such a hand can perform tasks that would be impossible with a two-finger gripper, such as using tools. Consequently, enabling a robot to perform the same tasks with an anthropomorphic hand that it could perform with a two-finger gripper increases its flexibility. Nonetheless, this also comes with additional challenges. An anthropomorphic hand has significantly more degrees of freedom (DOF) and is, therefore, more difficult to control.

One prominent approach is to use reinforcement learning (RL) to control these multi-fingered hands [4]. RL allows one to let the robot learn how to control the hand itself instead of having to decide how to move every joint of the hand manually. Additionally, it allows adaptation to varying circumstances, such as different objects for the handover. The second aim is, therefore, to train an RL model that performs the object-grasping part of the handover pipeline. Specifically, the thesis investigates the possibility of training such a model entirely on a real-world system without a simulator. As this latter part is an unusual approach, the following section further explains the motivation for this choice.

1.1. Real System Reinforcement Learning

As reinforcement learning is notoriously sample inefficient, the question arises about why it should be directly deployed on a real system instead of a simulation, as is typical. First, recent developments in RL, such as more sample-efficient algorithms, start to combat this challenge of training robots in the real world [5]. Allowing training on real-world systems also prevents one persistent issue for simulation training. This issue is the so-called sim-to-real or reality gap. It describes the difficulty of a simulation-trained system functioning in the real world. The reason for this is the simplified assumptions for models in a simulator. A simulator can not simulate the noise of a real system, i.e., calibration errors or motor slippage. Nor can a physics model calculate every interaction happening in the real world. When confronted with these noise sources, systems perform significantly worse than in simulation. While, for example, domain randomization options exist to combat this issue, they can only partially solve this problem in practice. Training directly on the system can prevent this issue entirely.

For the setup in this thesis, using the real robot directly is specifically beneficial. It uses a Willow Garage PR2 [6], which is a highly compliant system. Resultingly, any motion of one manipulator on an object held by the other manipulator, such as in a handover, can and will move the other arm slightly. It would be a challenge to simulate this behavior correctly. Further, the custom combination of the PR2 system with the anthropomorphic hand, a Shadow Dexterous hand [7], introduces additional difficulties. Such a composite system is challenging to calibrate correctly, as no standard implementation can be applied. Instead,

each system needs individual calibration. Combined, it results in a system with significant contributions to the sim-to-real gap besides the usual issues.

Finally, implementing such a learning system on a real robot can provide use even when a similar, simulation-trained model should emerge. Because of the difficulties described above, any such model will need help with deployment on the actual system. To compensate for this, such a pipeline to further train the model on the actual system could be a viable option.

1.2. Outline

The structure of this thesis starts with chapter 2 explaining fundamental concepts for the rest of this work. These involve robot kinematics, a robot's workspace, metrics for grasp stability, the hand synergies concept, and an introduction to reinforcement learning, including an explanation of Soft Actor-Critic, the reinforcement learning algorithm used in this thesis. Afterward, chapter 3 investigates related work in respective research fields. These are grasping approaches for anthropomorphic hands, reinforcement learning on real-world robot setups, and existing implementations for bimanual handovers. Next comes chapter 4 explaining the implementation of this approach. This explanation comes in three parts. They cover the used robot setup, the structure and individual components of the handover pipeline, and the reinforcement learning process. Following is an explanation and evaluation of the experiments performed with this setup in chapter 5. The two experiments of this thesis involve the mentioned workspace analysis for different grasp types, as well as training reinforcement learning models and testing them by executing the pipeline with them. Finally, chapter 6 discusses the results of the experiments and the pipeline structure in general, ending with the conclusion to this thesis in chapter 7.

2. Fundamentals

This chapter covers relevant fundamental concepts for this thesis. It starts by introducing robot kinematics, which is relevant to the control of the robot used in this thesis. Next comes an introduction to the concept of a robot's workspace, which is relevant for workspace analysis in a later experiment. Following comes the concept of grasp stability, which is part of the handover pipeline and the grasp training process. Afterward, this chapter covers the concept of hand synergies, which is relevant for controlling the anthropomorphic hand of the robot setup with the learned grasping model. Finally, the last section covers the basics of reinforcement learning and the specific algorithm used for training in this thesis.

2.1. Robot Kinematics

Robots allow for motion per design. Consequently, it is necessary to model the motion capabilities of a robot, as it allows plan and control of the robot's movements. Attempts to do so while ignoring the forces and torques applied to the robot fall in the field of kinematics [8].

Regarding this thesis, the most essential part of kinematics is the analysis of the serial chains of a robot, e.g., a robot arm. Such a chain consists of links, typically assumed to be rigid bodies that connect at joints. Each link connects to two others, except the first and last ones, which connect to only one other link. The first link's unconnected end is usually assumed to be mounted at a static position and called the base. Correspondingly, the end of the last link is called the end-effector, which, for robot arms, would be the point capable of manipulation, such as a gripper. Figure 2.1 depicts an example of such a setup. Therefore, the kinematic description of such a chain is called a kinematic chain.

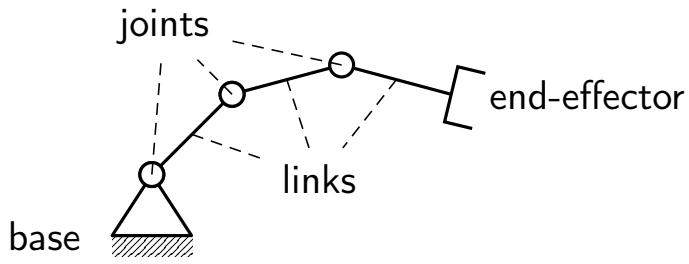


Figure 2.1.: Example schematic of a 2D robot arm with 3 joints.

A typical tool for describing such a kinematic chain are Denavit-Hartenberg (DH) parameters [9], which describe the relationship between two joints. These parameters are the link

length, link twist, link offset, and joint angle. Link twist and offset are usually fixed parameters, while the link length is variable for prismatic and the joint angle for revolute joints, the two most common joint types in robotics.

There are two parts to describing robot kinematics: Forward and Inverse Kinematics. The straightforward part is Forward Kinematics (FK) [10]. It describes the problem of mapping a joint configuration for a robot's kinematic chain from a base frame to the corresponding pose of an end-effector in cartesian space. A pose in this context means the position and orientation in cartesian space. Using these parameters, applying a transform from one joint to the next for each joint in the kinematic chain can compute the transform from the base to the end-effector. In an example with a kinematic chain of seven joints, as is the case for the arms of the robot used in this thesis, such a transform chain would be

$$T^{\text{base}}_{\text{end-effector}} = T^0_1(q_1)T^1_2(q_2)T^2_3(q_3)T^3_4(q_4)T^4_5(q_5)T^5_6(q_6)T^6_7(q_7) \tag{2.1}$$

with an transform T^{i-1}_i of joint variable q_i for joint i. In this case, q_i describes either the link length or the joint angle for a prismatic joint or revolute joint in the DH description model, respectively. Writing these transforms as

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

where p_x, p_y, p_z are the values of the position vector and r_{kj} are the rotational values of the transform, allows calculating these values from the DH parameters of the joint combined with the corresponding joint variable q. The resulting transform can be computed and applied to the base to get the deterministic end-effector pose for the provided joint configuration.

Inverse Kinematics (IK) describes the problem of generating a joint configuration for a robot's kinematic chain for a given end-effector pose in the cartesian space. In that sense, IK maps a cartesian pose to a corresponding pose in the robot's joint space. Contrary to FK, the IK problem is significantly more challenging to solve. Generally, the two types of strategies to solve IK are analytical and iterative methods. As with FK, the first step to analytically solving IK is to create a chain of transforms from the base to the end-effector. This time, however, only the resulting transform is known through the provided goal pose of the end-effector. Each other transform contains the unknown rotation and translation variables between one joint and the next.

Considering the same example as previously, by transferring the first transform to the left side of Equation 2.1, this side now only contains the known values of transform $T^{\text{base}}_{\text{end-effector}}$ and values depending on q_1. In contrast, the right side only depends on the values of all remaining joint variables $q_2, ...q_7$. Using the representation in Equation 2.2 results in 12 equations to be solved. Suppose this gets repeated with the second transform. In that case, it results in 12 additional equations, so together with Equation 2.1, this example would produce 12 sets of seven non-linear equations, one set for each rotational and translational element. Each equation set must be solved to get possible solutions for the IK problem.

Analytically solving IK provides all possible solutions and determines if solutions exist. Additionally, once all required equations are derived, calculating new solutions becomes fast. However, it is usually challenging to derive all equations, which must be derived for each robot independently. On the contrary, iterative or numerical approaches aim to find sequences of joint variables that minimize the error between their resulting end-effector pose and the desired pose through iterative optimization. Consequently, these approaches are applicable independently of the robot's kinematic structure. However, they are more computationally expensive and may get stuck in local minima, never finding an optimal solution. This results in the need to check any solutions afterward again with FK. Nonetheless, most applications use these iterative solutions, as does this thesis.

2.2. Workspace

As defined by Gupta [11], a robot's workspace consists of all reachable points by the corresponding end-effector. A robot arm's workspace is defined relative to its base or shoulder, depending on whether it is fixed or mounted on a robot torso, and contains all reachable end-effector poses for the entire arm. Kumar et al. [12] further differentiate between the reachable workspace, which follows the previous definition, and the dexterous workspace. The dexterous workspace is a subset of the reachable workspace and describes every point any desired end-effector orientation can reach.

For a bimanual robot, the workspace can also be reported as the combined workspace of both arms if a task only requires reaching a position with one end-effector. However, information about all reachable positions is often only one of the required information. Tasks involving a robot's end-effector commonly also require a specific orientation. Further, some joint configurations are more desirable than other viable configurations if they are less restrictive to further arm movements after reaching it has reached the goal. Consequently, operating in the dexterous workspace of both arms would be ideal. However, these do not always have an intersection, so bimanual tasks often have to find a part of the reachable workspace that is suitable for the task.

To assign a measure of desirability to these configurations, Yoshikawa [13] presented a manipulability measure. It assigns a numeric value to each viable joint configuration based on its distance to singular configurations. This measure has been extended further, e.g., by Vahrenkamp et al. [14]. They added further penalizations based on closeness to joint limits or obstacles in the environment or the robot itself.

Since a robot's workspace is continuous, analyzing the quality of different workspace parts commonly involves using a discrete grid. Such a grid is the basis for identifying the optimal region to perform a task, e.g., the best region to sample grasps. One such approach can be seen in the work by Zacharias et al. [15], where a capability map for the workspace of their robot gets computed based on the number of valid IK solutions in each region for different end-effector orientations. Such a map then allows for finding a suitable robot configuration to perform a task, such as grasping a bottle. Due to the number of IK computations required to perform a workspace analysis and, for example, create such a contact map, computing such an analysis is done offline. Zacharias et al. reported a computation time above 12 h. Even though

they mention having an unoptimized algorithm, this duration shows this necessity. During online task execution, only the best options, according to the analysis, are then chosen.

2.3. Grasp Stability

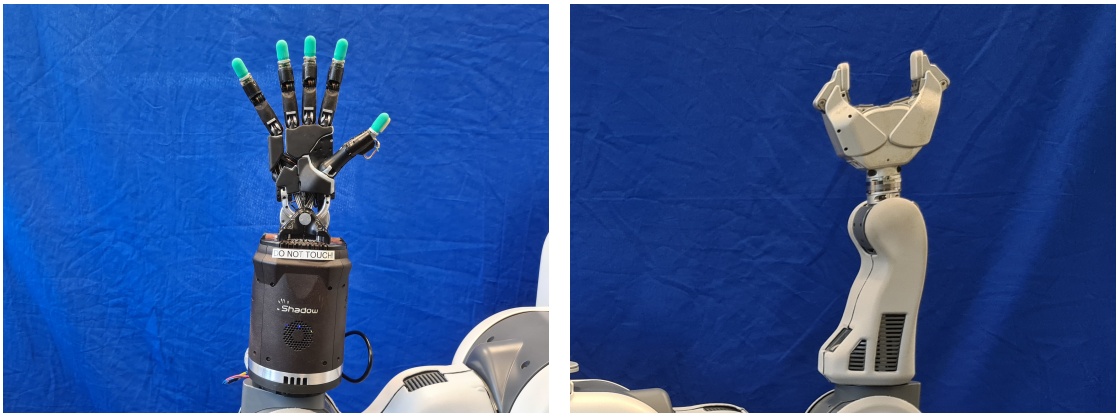


Figure 2.2.: The left image shows the multi-fingered hand used as one manipulator in this thesis, while the right image shows the two-finger gripper used as the other.

During a bimanual handover, grasping an object is an essential part. Consequently, finding a grasp for the robot and evaluating its stability is also essential. In this context, a grasp means a configuration for a robot manipulator where the manipulator makes contact with the object, allowing the manipulator to hold the object. The robot manipulators relevant to this thesis are two-fingered grippers and multi-fingered hands, seen in Figure 2.2. As the names suggest, two-finger grippers have two movable fingers that are typically opposite to each other. This configuration allows them to exert forces on an object opposing each other.

Conversely, multi-fingered manipulators have multiple movable fingers in various configurations. However, multi-fingered hands are a specific subgroup of these multi-fingered manipulators that aim to have their fingers in a configuration similar to a human hand. Consequently, these typically have a thumb and some parallel fingers. Especially for this last category, it is challenging to find grasps that manage to hold an object. Therefore, it is essential to have a method to evaluate the stability of different grasp candidates to allow for the selection of a suitable candidate.

There are multiple ways to predict grasp stability for multi-fingered robotic hands. A typical way is to consider the geometric properties of a grasp, such as the grasped volume. Further, the properties of the grasp can be analyzed based on the hand configuration, e.g., analyzing the distance to the joint limits for all finger joints. A list of methods in these categories was composed by Roa et al. [16]. One way is to measure the contact forces of the fingers on the object. The goal is to ensure the hand applies enough force so enough friction can be applied to prevent the object from slipping. This option is called slip-detection, as in work by Zapata-Impata et al. [17]. They present the two types of slippage, translational and

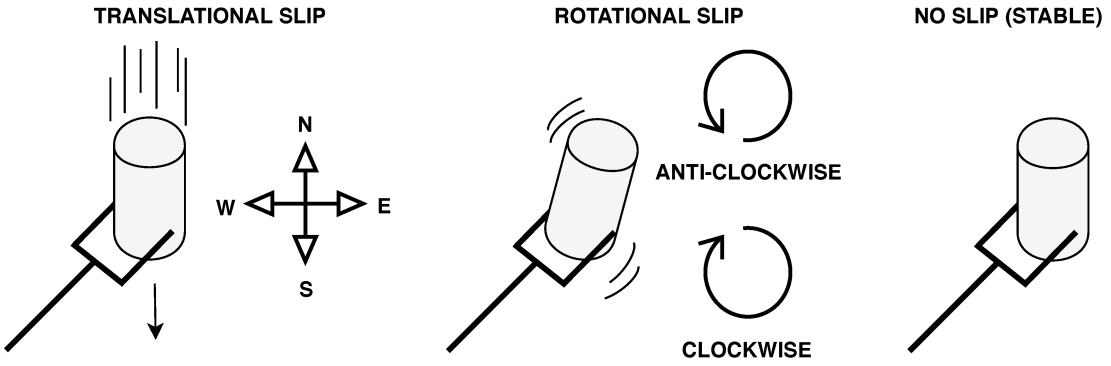


Figure 2.3.: The three options of slippage during robot grasps: translational, rotational, or no slip. [17]

rotational, as seen in Figure 2.3.

In their work, the authors present a tactile-based deep-learning approach to detect possible slippage. An overview of other techniques, such as friction, vibrational, or optical-based techniques, can be found in the survey by Romeo et al. [18]. This thesis uses a simplified friction-based approach to decide on grasp stability through slippage detection.

This detection method aims to prevent translational slippage. As seen in Figure 2.4, the fingers apply a normal force F_n to the object when grasping, while a tangential force F_t also applies through gravity. With the simplified Coulomb friction model, it is stable if

$$\mu_S \geq \frac{F_t}{F_n} \tag{2.3}$$

where μ_S is the static friction coefficient of the object.

It exploits the bimanual nature of grasps in the used setup to circumvent the limitation mentioned by Zapata-Impata et al. of manually labeling successful grasps. For more information, subsection 4.2.4 provides a more detailed explanation.

2.4. Hand Synergies

Human hands are versatile and complex natural manipulators. As robotics advanced, the desire arose to utilize similar capabilities for robots. This advancement resulted in the creation of anthropomorphic robot manipulators. While they have evolved into capable manipulators resembling human hands, such as the Shadow Dexterous hand [7], these high DOF manipulators now face the issue of controlling them to perform tasks such as object grasping. To solve this issue, researchers again took inspiration from us humans. One controlling concept arising from this research is the concept of hand synergies.

First discovered by Santello et al. [20], we humans do not operate our hands by controlling each joint independently. Instead, we control them in a lower DOF space. They recorded a dataset of human grasps by recording the hand poses of humans. Utilizing principal component analysis (PCA), the authors then investigated how much each principal component

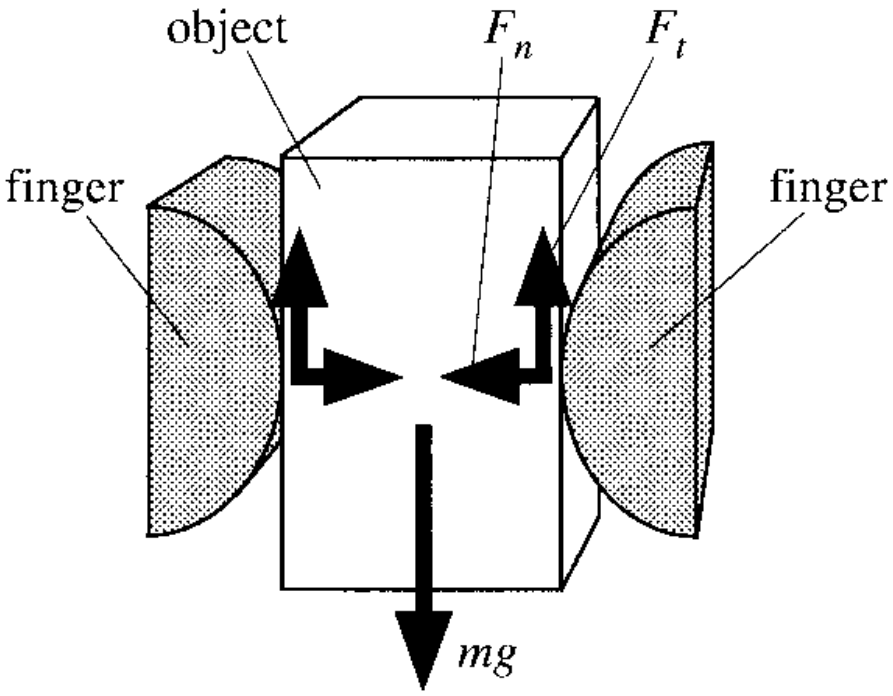


Figure 2.4.: Simplified model of acting forces during an object grasp. [19]

contributed to the variance in the data. They found that the first two components could already explain over 80% of the variance.

This potential of controlling a complex, high DOF manipulator in such a low-dimensional space caught the interest of roboticists, and attempts to utilize this concept in robotics emerged. Ciocarlie et al. [21] directly used the data collected by Santello et al. They mapped the values for each finger joint of the human hand to corresponding joints for various robotic hands. They derived two hand posture subspaces for each robotic hand, which they referred to as eigengrasps. Proceeding, they used these eigengrasps to generate new grasps for each hand. Generating new grasps can be done by adding all eigengrasps, with a corresponding weight for each, and adding the corresponding vector to an origin posture. Specifically, Ciocarlie et al. defined any posture possible with their eigengrasps through

$$p = p_m + \sum_{i=1}^{b} a_i e_i \tag{2.4}$$

with p as the resulting hand posture and p_m as the origin or default posture. With b as the number of eigengrasps, e_i is the eigengrasp i, and a_i is the corresponding weight assigned to the eigengrasp. Therefore, one can describe a hand posture through only a vector $a = [a_0, ..., a_b]$ instead of specifying values for each robotic hand joint.

Consequently, it is possible to significantly reduce the dimensionality of controlling an anthropomorphic hand if such an eigengrasp or synergy dataset is available for the used hand.

Liang et al. [22], and Bernardino et al. [23] already generated such a dataset for the same type of robotic hand as the one in this thesis in their previous works. They made the data available for use in this thesis. The remainder of this thesis will relate to the posture subspaces or eigengrasps as synergies or hand synergies.

2.5. Reinforcement Learning

Most task assignments for a robot involve interacting with a complex environment to achieve its goal. Manually programming a robot to always perform the best actions in every scenario is often impossible. It would require perfect knowledge of all possible states of the environment and the robot, which are unknown even to humans in uncontrolled scenarios such as moving around in the real world. If robots ever are to be used in the real world, solving this issue is critical. Since humans and animals manage to work in the real world just fine without perfect information, researchers investigated how we learn to operate in such an uncontrolled environment.

Reinforcement Learning (RL) is an approach to implement a computational version of this learning process [24]. As can be observed by infants, the learning process involves repeated interaction with the environment and evaluating the response from the environment, given that action. For living creatures, the most simple rewarding responses are pleasure or pain, depending on whether the action was successful or detrimental. RL tries to emulate this simplified concept, reducing the process to an *agent* interacting with an unknown *environment*. More specifically, the agent performs an *action* and then observes how the *state* of the environment changes and what *reward* the chosen action produces. Figure 2.5 illustrates this simplified process. The goal of this learning process is now to find a decision process to choose the best actions with the information provided by the environment.

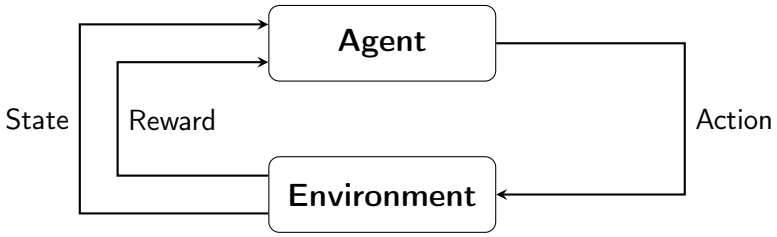


Figure 2.5.: Simplified model of the relationship between an agent and its environment in the context of RL.

To apply this concept to the overlying problem, the agent must utilize the given information and decide which action to choose. A standard model for this decision process is a Markov Decision Process (MDP). It models the process in discrete timesteps t, where the agent starts in the s_t, one state s at time t. Based on the action a it decides to perform at time t, namely a_t, it then transfers to state s_{t+1} and gets the associated reward r at that timestep, r_{t+1}. Each state also has to follow the *Markov property*. This property states that each state s_t must depend only on the previous state s_{t-1} and the previous action a_{t-1}. Consequently,

each state has to include all relevant information to decide on an action in the future. Such a restriction allows the training to only consider the current state at each timestep instead of evaluating all previous states.

With that, the goal is to find a *policy*, which decides what action maximizes the discounted reward over the episode. Usually, this policy represents a function assigning a probability to each action a given the state s, commonly written as $\pi(a|s)$. Any algorithm generating such a policy must also balance the concepts of *exploitation* and *exploration*. Ultimately, the goal is to find the policy that chooses the actions resulting in the best rewards. This idea is the concept of exploitation. However, suppose the agent would only focus on exploiting its learned relationship between actions and rewards. In that case, it might miss out on better action it has not sufficiently tested. Testing actions that are not optimal is the concept of exploration. Neither concept can be solely applied to any agent, as an entirely exploitative agent would most likely not find the best actions, and a fully explorative agent is incapable of learning a suitable policy. There is no optimal solution to this conundrum, and finding a balance between both concepts is essential for any algorithm learning a policy.

RL algorithms describe how the policy should change concerning past experiences. Most algorithms utilize a *value function* to derive such a policy change. The value function is a numerical representation of the value for any state, then named a *state-value function* or state-action pair, noted as an *action-value function*. Such a value is usually calculated through the *expected return*, the expected value of all future rewards. By using the expected return instead of just the next expected reward, the value considers the possibility of performing worse actions in the short term to achieve better rewards in the long term. Formally, one can write the return at timestep t as

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{2.5}$$

with T as the episode length and the expected return the corresponding expected value. Here, $\gamma \in [0, 1]$ is the so-called *discount factor* to value short-term rewards higher than rewards expected later, ensuring the expected reward to be a finite value even for potentially infinite models. Resultingly, one can note a state-value function as

$$v_\pi(s) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right], \forall s \in S \tag{2.6}$$

and an action-value function as

$$q_\pi(s, a) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right] \tag{2.7}$$

for a policy π. As the real result distributions are unknown, the agent has to estimate these value functions based on the rewards it has already experienced. An alternative representation of the value function is

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s') \right], \forall s \in S \tag{2.8}$$

Here, s' is a successor state to state s, and $p(s', r|s, a)$ is the probability of receiving reward r and arriving at state s' when selecting action a at state s. The Equation 2.8 is called the *Bellman equation*. It is essential for many RL algorithms, as it allows the expression of the value function as a recursive relation between a state and all its successor states.

One can now evaluate different policies by their value function. If a policy's value function provides better values for all states than the value function of another policy, it is a better policy to use. Theoretically, finding the optimal policy can be done by finding the policy with the best value function through optimizing all policies.

While this simplified assumption is generally the foundation of RL problems, one further attribute splits these problems into two categories. This attribute is whether or not a *model* of the environment exists for the agent. This model does not need to represent the environment perfectly but should either try to mimic the environment or allow for predictions of the behavior of the environment. Such a model would allow an agent to plan actions to some extent. Methods relying on such a model are called *model-based* methods. On the contrary, *model-free* methods do not require such a model and try to solve the problem through trial and error. In the following, the focus lies only on model-free approaches, as Valencia et al. [25] have shown that these are better suited for the tasks in this thesis, as later explained in section 3.2.

More realistic RL tasks typically do not fulfill the previously explained fundamentals. Analytically calculating a value function is often impossible, not to mention optimizing all possible policies about their value function. Modern RL algorithms, therefore, approximate the policies and corresponding value functions. However, the available RL algorithms are generally split again into two additional types, *on-policy* and *off-policy* algorithms.

On-policy algorithms try to optimize the policy with which they collect their data. In theory, they converge faster to their optimal policy. However, this comes with the cost of a suboptimal policy, as it always has to balance exploration and exploitation to ensure new samples can be collected.

Off-policy algorithms, on the other hand, use two policies. One exists for data collection with a stronger focus on exploration. The other is the policy optimized regarding exploitation. While off-policy algorithms often converge slower, they are considered more potent. Further, modern off-policy algorithms often have mechanisms to reuse samples, making them more sample-efficient. For this thesis, the chosen algorithm is SAC, an off-policy algorithm. It gets further explained in subsection 2.5.1.

2.5.1. Soft Actor-Critic

SAC [26] stands for Soft Actor-Critic and is an off-policy RL algorithm. As the name suggests, the algorithm uses an actor-critic architecture. Such an architecture learns two models simultaneously. One network, called the actor, decides which action to take at a current state and represents the current policy. The second one, the critic, represents the value function and provides the estimated values for each state. Both networks train simultaneously, where the actor learns the optimal policy relative to the critic's values, and the critic learns the values of the states reached by the actions the actor decides.

The "Soft" part in SAC implies using a softmax function for the actor to decide the following action. This decision transforms the actor into a stochastic actor, choosing actions based on a probability distribution instead of the deterministic action that maximizes the currently expected reward.

Further, SAC expands the maximum expected reward objective by an additional entropy term. This entropy term describes how much information different actions provide from the given state. By aiming to maximize the long-term entropy as well as the expected reward, the agent is encouraged to increase exploration. One can compute the entropy values from the probability distributions of the actions. The entropy parameter, also referred to as the entropy coefficient or through its inverse as a reward scale, regulates the influence of the entropy term.

Additionally, SAC employs a method called double Q-learning. It involves training two separate Q-functions, the functions that estimate the value function, and using the lowest value of both functions to calculate the gradients used for updating the actor and value functions. In practice, this measure allows for faster training. Overall, Haarnoja et al. [26] describe their algorithm SAC as relatively resistant to hyperparameter changes, except for the entropy coefficient, and sample efficient, making it a suitable choice for this thesis.

3. Related Work

This chapter covers previous work in related fields. Object grasping is an essential concept for bimanual handovers. As the robot used in this thesis has an anthropomorphic hand, the chapter starts with a section about object grasping with anthropomorphic hands. Afterward, it provides an overview of existing work that used reinforcement learning in a real-world setting. Finally, the chapter covers other implementations of bimanual handover pipelines.

3.1. Grasping

Most recent approaches use RL as part of their manipulation approach. Yu et al. [4] provide an overview of RL methods applied to various dexterous manipulation tasks, including object grasping. They present different multi-fingered manipulators and review applications using these about their training methods, such as training from scratch or demonstration. The authors find that the typical challenges for RL, such as sample inefficiency and the reality gap, still exist, and several complex tasks regarding manipulation remain unsolved. Consequently, the aim of this thesis to circumvent the reality gap remains supported by their findings.

Figure 3.1.: An example from the work of Liang et al. [22] of their trained model grasping a bleach bottle.

In one recent work, Liang et al. [22] utilize the same hand as the setup used in this thesis. They utilize grasp synergies to reduce the required dimensionality for hand control significantly. With these grasp synergies, they train a neural network using reinforcement learning. They generate an initial grasp configuration and refine the grasp using the network. This results

in the ability to grasp various objects of different object instances, as seen in Figure 3.1. However, they only test this approach on an unimanual system. Further, they trained their model in simulation, so transferring the ability to train a similar model on the real robot still needs to be investigated.

Another approach by Brahmbhatt et al. [27] performs grasp synthesis based on optimization for contacts. They use contact maps generated by human demonstration to create optimized grasps to simulate these contacts. For grasp generation, they use the popular GraspIt! simulation. The contact database splits the contact maps into two categories: one for object use and one for object handover. Therefore, their algorithm can generate grasps for both cases as well. Their tests show that the approach works with different hand models. However, the testing was only done in simulation and not on real hardware. Further, generating contact maps for novel objects would require significant additional work, making this approach difficult to expand to be able to generalize.

The paper by Li et al. [28] explores a grasp generation method based on Gaussian processes. Since correctly identifying the shape of an object during grasping is very difficult, their probabilistic model aims to take that uncertainty into account for grasp generation. By generating an object model from the detected point cloud of an object, their model samples several grasp point candidates from the object. An optimizer then aligns the targeted grasp point with a virtual grasp frame designed for the hand to get the desired grasping configuration. During grasping, the focus shifts to a closing control scheme focusing on the measured contact forces. However, this approach requires prior knowledge of the object to generate the online model from the point cloud. Further, they test this approach only on hands with significantly lower DOFs than the Shadow hand.

In their work, Roa et al. [29] present a grasp planner focused on generating power grasps to hold an object. Sampling through slices of the object first allows for selecting the best candidate region that enables the circular finger configuration required for a power grasp. Afterward, the algorithm samples the contact point for the palm. Next, contact points with as many fingers as possible are generated through a sequential closing strategy of the fingers, resulting in a power grasp on the object. Besides focusing on only power grasps, models of the objects and the hand are required for this grasp generation. Such an approach would require modeling each new object before grasping it, contrary to the approach in this thesis.

Shao et al. [30] present a grasp generation approach that aims to be gripper-independent. By combining a gripper's URDF file with the point cloud readings of the gripper, they train an autoencoder to produce a model of the gripper used. They then use the autoencoder to extract features for the gripper representation. Together with features extracted from the point cloud of the object, they use it as input for another neural network. This process generates contact points for each finger of the gripper on the object surface, which describes a grasp with enough force closure. While the authors claim this process applies to anthropomorphic hands, they only trained with two- and three-finger grippers. Even though they performed experiments with anthropomorphic hands, they manually limited the number of used fingers for these hands to three fingers, so applying this approach to hands with significantly more DOFs will most likely prove to be complicated.

In another approach, Aktas et al. [31] present a deep learning approach to generate grasps

from a single object view. When viewing the object, the model uses the generated point cloud as input for a generative model to generate possible grasp candidates at this point cloud. They train the model in simulation based on demonstrated grasps. It generates grasp configurations for anthropomorphic hands by reproducing the recorded contact points from the demonstrated grasp. To improve the generated grasp's success, they train an evaluation model to predict if the generated grasp will be successful. This model is trained in simulation as well. They experiment with different models for the generative and evaluative parts in simulation and on a real robot.

3.2. Real Robot Reinforcement Learning

Real-world training scenarios for reinforcement learning are rare in robotics. Due to sample efficiency, hardware damage risk, and increased training time, most approaches prefer simulations to real robot deployments. However, with more modern reinforcement learning algorithms, some work exploring this possibility has emerged, as it provides a possible solution to the prevalent reality gap.

Dulac-Arnold et al. [32] provide an overview of the challenges regarding real-world RL. It lists nine challenges, including the limited number of available samples and the high dimensionality of the continuous state and action spaces. They also propose an evaluation method for each challenge to compare various approaches. To this end, they also propose a benchmark to evaluate the varying RL algorithms. This thesis focuses on solving the two named challenges.

In their work, Mahmood et al. [33] benchmark several reinforcement learning algorithms directly on real-world tasks, including reaching specific positions, docking to a loading station, and tracking motions. They investigate four reinforcement learning algorithms, TRPO, PPO, Soft-Q, and DDPG, on tasks employed on a robotic arm and a mobile platform. Their focus is to determine the sensitivity of hyperparameters for the different tasks. While encountering severe challenges, including a very long run time of the experiments of over 950 hours, they prove the possibility of employing reinforcement learning algorithms directly in real-world scenarios. Unfortunately, this benchmark did not investigate the reinforcement learning algorithm used in this thesis, SAC.

Testing SAC for real robot applications was done by Haarnoja et al. [34], including some of the original authors of the SAC paper. They employ it in two real-world scenarios: movement for a quadrupedal robot and dexterous manipulation by a robot hand. They managed to solve both tasks without using any simulation of their environment. Further, they also compared the performance of SAC with PPO on the manipulation task and found SAC to significantly outperform PPO in terms of training time while succeeding on the task. These findings show that SAC might also suit this thesis's scenario.

Further research by Valencia et al. [25] compares model-free and model-based reinforcement learning approaches in real robot manipulation scenarios. The learned tasks involve turning a valve and manipulating a cube. They compared the efficiency and performance of both kinds of RL. Results show a clear advantage of the model-free reinforcement learning approach compared to the model-based one, even with a well-performing model. The authors explain these findings, which contradict previous research, with the difference between training in

simulations and on the real robot. Points of failure in the real world, such as friction and signal noise, apparently disrupt the model-based training significantly. These findings encourage a model-free approach for real-world reinforcement learning, such as the one in this thesis.

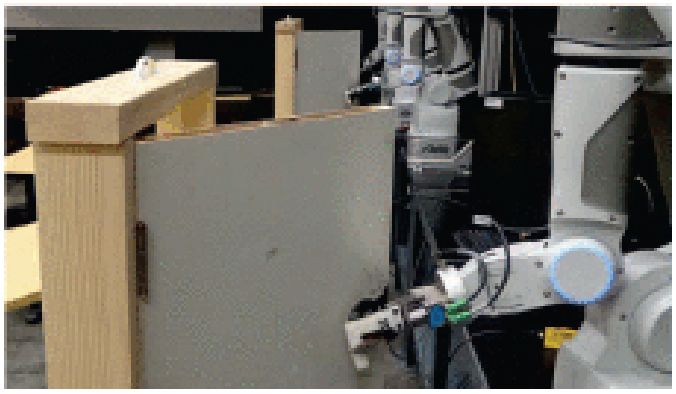


Figure 3.2.: The training setup of Gu et al. [35] for training their robot to open doors in the real world.

Gu et al. [35] propose an off-policy reinforcement learning algorithm utilizing asynchronous updates from multiple agents. They let their agents learn to reach for and open doors, as seen in Figure 3.2. The agents train this complex task in simulation and the real world. Their results show that real-world training allows their agents to successfully learn this complex task in a reasonable time of 2.5 - 4 hours. However, having multiple agents of the same type of train in the real world can be difficult, especially with more complex robot setups like the one used for this thesis. Nonetheless, their work shows that even a single agent can learn complex tasks in the real world with modern off-policy reinforcement learning algorithms, such as the one employed in this thesis.

3.3. Bimanual Handover

Performing bimanual handovers is a specific subtask of the field of dual-arm manipulation. Smith et al. [2] provide an overview of this general field. First, they compare different dual-arm robotic platforms, including the PR2. Afterward, they review different approaches regarding their setups and concepts. These include comparisons regarding system modeling, grasping strategies, and learning strategies.

A recently published approach by Li et al. [36] focuses on bimanual handover. They use two similar Franka Emika Panda industrial arms to perform multiple handovers of block-shaped objects. In their implementation, they learned a grasping policy utilizing reinforcement learning. They employ the symmetry of their arms to enhance the efficiency of their training. By mirroring the learned SAC policy on one arm to the other, they learn a better-performing policy more efficiently. This thesis cannot apply this strategy due to the difference in both manipulators and their use of only two-finger grippers.

Saut et al. [37] propose a planner capable of planning object pick-and-place tasks involving

the bimanual handover of the object. Their planner can work with a humanoid upper-body robot with two anthropomorphic hands. The planning process involves an analytical process for the grasp planning instead of a learned grasping policy. It calculates possible grasps by computing possible intersections of the object surface with the workspaces of the fingers on each hand. These grasps are computed offline for faster computation of possible grasping solutions. Bimanual grasps are computed by comparing lists of grasps for each arm and filtering and scoring all possible combinations to find a solution. However, this approach requires a scanned point set of the object beforehand to generate these grasps, adding additional work for adding new objects. Further, the authors tested this approach only in simulation with one object, contrary to the approach of this thesis.

In contrast to most work in the field, Cruciani et al. [38] define a planner capable of bimanual re-grasping through a dexterous manipulation graph. Their planner derives the graph from a point cloud reading of the manipulated object. It contains possible configurations of the gripper along the object surface in its nodes and movement between two configurations along the object surface as its edges. This graph combines the capability of in-hand manipulation with bimanual re-grasping if the desired in-hand manipulation is impossible in the current configuration. It can then generate possible grasps for the second gripper to grasp the object while the first gripper still holds it. While allowing for complex re-grasping procedures, the process is only designed and tested for two two-finger grippers, contrary to the approach of this thesis.

Another implementation, which allows bimanual handover, can be found in the work by Haschke et al. [39]. The authors implement a geometry-based grasping pipeline for a bimanual Shadow hand setup. While the focus is to solve pick-and-place tasks, it is capable of performing a bimanual object handover if required for the task. It utilizes geometric information about the objects instead of a deep learning approach. Modeling the object's rough shape uses super quadrics on a segmented point cloud. Different possible grasp candidates for this shape are evaluated based on a previously defined grasping frame for the hand. Planning the different steps for pick, place, and handover uses the MoveIt task constructor. However, their grasp candidates are limited to fixed, previously chosen grasp types.

One paper published by Vezzani et al. [40] works with a humanoid upper body with two anthropomorphic hands. They present a pipeline to perform a stable initial grasp and handover. They successfully hand over various objects using grasp stabilization with a three-finger precision grasp with tactile feedback. Similar to this thesis, they gather information about the objects through point cloud filtering. However, their grasping pose selection relies on a priori assigned grasping poses for each object, requiring prior object modeling to generate these poses. This process requires significant work for each object, making it difficult to add new objects.

In another publication, Balaguer et al. [41] propose an algorithm for bimanual in-air regrasping utilizing a supervised machine learning algorithm. They generate initial grasping points on an object through image preprocessing on a stereo image. Together with a point cloud of the object, they use the preprocessed image and the initial grasp points as input for a machine learning algorithm initially developed for unimanual grasping. The generated configurations for the grasps of both manipulators are then further optimized to find the most

efficient handover configuration regarding execution time. This approach has two significant downsides compared to this thesis. First, the authors only used hands with four degrees of freedom and did not consider the complexity of performing a grasp with a hand with higher degrees of freedom. This issue makes it not applicable to anthropomorphic hands. Second, they rely on human-annotated data to train their machine-learning algorithm. Extending it for further object types not in their datasets takes significant time of manual data collection beforehand, unlike the approach in this thesis.

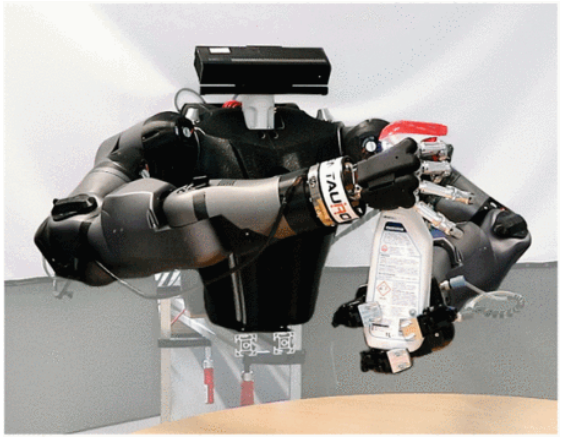


Figure 3.3.: An example for a functional re-grasp from the work of Pavlichenko et al. [42].

One paper utilizing a similar setup to this thesis, published by Pavlichenko et al. [42], uses a humanoid upper body equipped with a three-finger gripper as well as an anthropomorphic Schunk hand. They use both arms to pick up an object with the three-finger gripper and perform a functional re-grasping with the other hand, seen in Figure 3.3. The re-grasping is planned by reshaping meshes of previously known objects of the same object class to match the point cloud perceived of the current object. This mesh reshaping also shifts the predetermined functional grasp pose to the new object. While this allows the performance of functional grasps on various objects of the same instance, the reliance on several meshes of objects of the same kind makes it difficult to broaden this approach to multiple objects of different types when functional grasps are not required.

Further, Vahrenkamp et al. [43] present their efficient IK-based motion planner. This planner can quickly plan joint configurations for single and double-arm object grasps. They test it in simulation and a real-world scenario on a bimanual setup with anthropomorphic hands. Similar to this thesis, the authors precompute a reachability space to speed up the process of generating their solutions online. However, they only tested their planners with one object with fixed grasp annotations.

4. Approach

This chapter covers the implementation of this thesis's approach. The first section of this chapter describes the robot platform used in greater detail. Afterward, the second section explains the implementation of the overall handover pipeline. Finally, the last section explains the implementation of RL training for a grasping model on the real-world system.

4.1. Setup

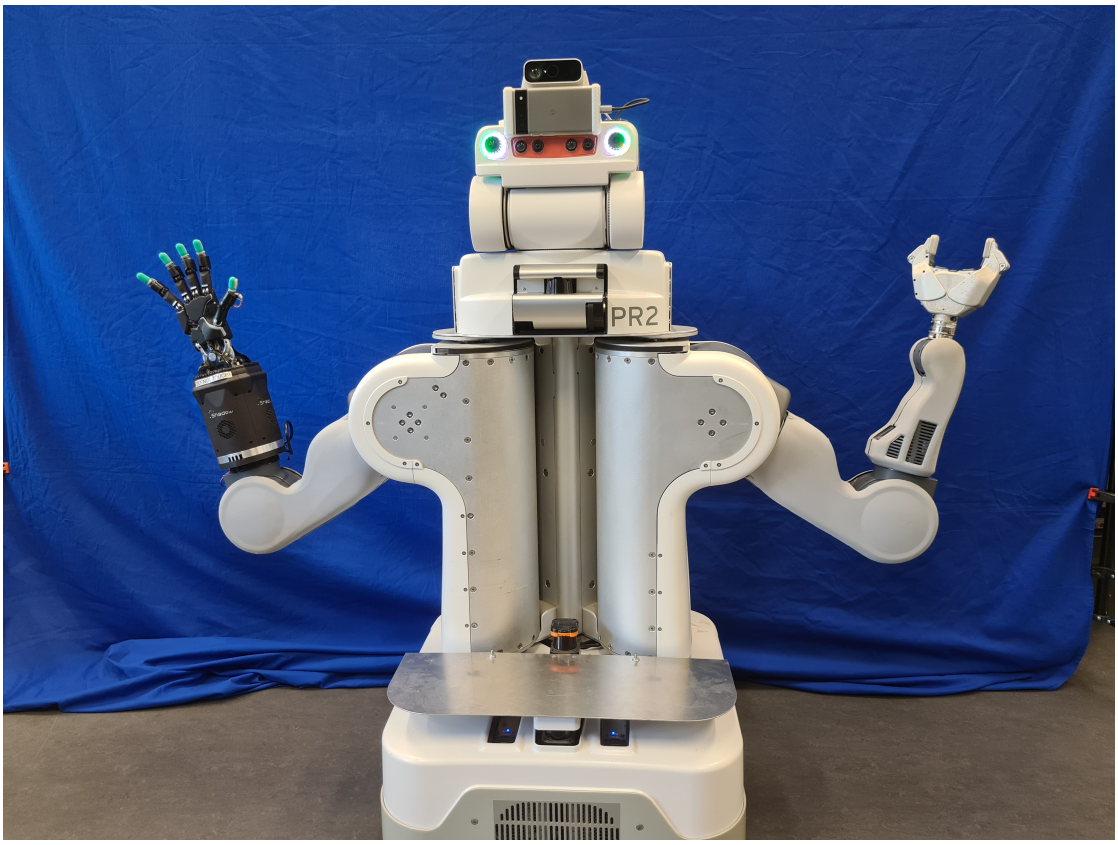


Figure 4.1.: PR2 robot with an Azure Kinect attached on its head and a Shadow Dexterous Hand.

The robot used in this thesis is a Willow Garage PR2 [6]. It is a bimanual, mobile robot platform. Figure 4.1 shows this modified version. Each arm has a corresponding kinematic

Table 4.1.: Joint limits for the joints of both arm manipulation groups.

Joint Name	Lower Limit	Upper Limit	Left Arm	Right Arm
Shoulder Pan	-32°	122°	✓	✓
Shoulder Lift	-20°	74°	✓	✓
Upper Arm Roll	-37°	215°	✓	✓
Elbow Flex	-122°	-9°	✓	✓
Forearm Roll	Continuous	Continuous	✓	✓
Wrist Flex	-115°	-6°	✓	✗
Wrist Roll	Continuous	Continuous	✓	✗
Hand Wrist J2	-30°	10°	✗	✓
Hand Wrist J1	-40°	28°	✗	✓

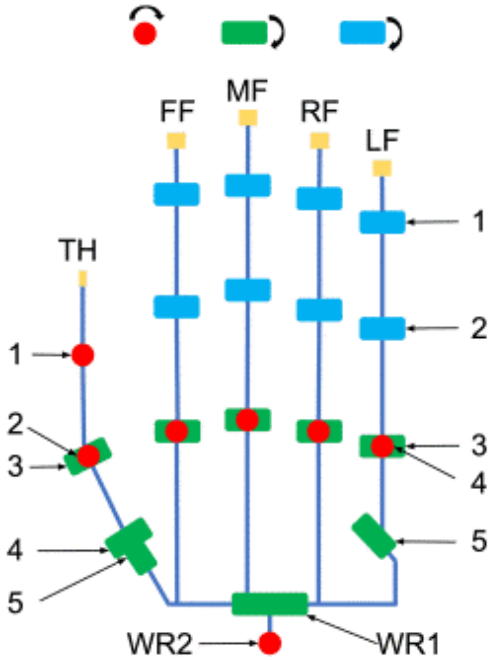


Figure 4.2.: Joint schematic of a left Shadow hand without BioTac sensors. [22]

group. Further, Table 4.1 lists each joint with its corresponding upper and lower limits.

As the first modification, the PR2 has a Microsoft Azure Kinect RGB-D camera mounted on its head. Additionally, it has its right lower arm and manipulator replaced by a Shadow Dexterous Hand [7] from the Shadow Robot Company. This hand has two wrist joints and usually 22 finger joints, as seen in Figure 4.2. However, the Shadow hand in this setup has its fingertips replaced with SynTouch BioTac tactile sensors. While it allows for tactile sensing with the fingertips, it also disables the first joints for all fingers, including the thumb, reducing the number of finger joints to 17. Two tendons control each joint by pulling the

joint to either stretch or contract. The force difference can be sensed through force sensors at each tendon and used as an effort measurement for each joint. In the wrist of the left arm with the two-finger gripper, the PR2 has a force/torque sensor installed.

This thesis utilizes the Robot Operating System (ROS) [44] as the underlying communication framework. Planning and robot control utilizes the MoveIt [45] framework. Any implicit IK calls use the default MoveIt IK solver. For explicit calls, this thesis uses BioIK [46], developed by Ruppel et al. It is a memetic algorithm utilizing various optimization methods that allow the consideration of different goal types for solving IK problems. For motion planning to get to the IK configurations, the pipeline also uses the default motion planner in MoveIt from the Open Motion Planner Library (OMPL) [47], Rapidly-exploring Random Trees (RRTConnect) [48].

4.2. Pipeline

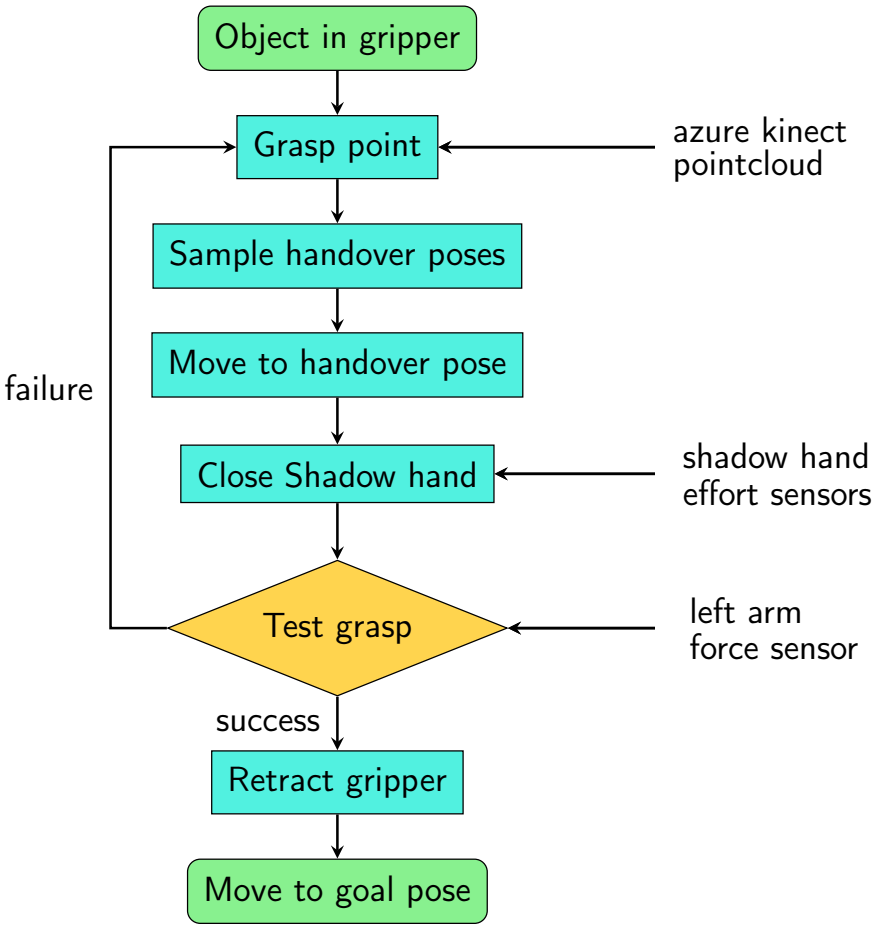


Figure 4.3.: Simplified structure of the handover pipeline.

Figure 4.3 depicts a schematic of the pipeline structure. This pipeline works with the following assumptions. First is the assumption that the handover object starts in the left gripper of the PR2. Second is the assumption that the object is in a stable grasp while held by the gripper. Correspondingly, the pipeline ignores any movement of the object during gripper movements. As all objects used are rigid, and several frameworks to perform stable grasps of objects with a two-finger gripper already exist, these assumptions are reasonable to assume, as it is not the focus of this thesis. Further, this handover procedure aims to move the object in an area that the left arm of the PR2 can not reach without the object falling. Any work done with the object afterward is not the focus of this thesis.

The pipeline only enables handovers from the two-finger gripper to the shadow hand. Since the robot has different manipulators on each arm, performing a handover in either direction has challenges in different areas from each other. When the object starts in the two-finger gripper, the object is more likely to be in a stable grasp without too much movement during the handover. However, controlling the anthropomorphic hand is significantly more challenging than controlling the gripper, which would be the case in the other direction. On the other hand, starting with the object in the Shadow hand makes object movement more likely, adding the challenge of in-hand object pose estimation. For this thesis, the focus is, therefore, to tackle the challenges of one direction.

4.2.1. Grasp Point

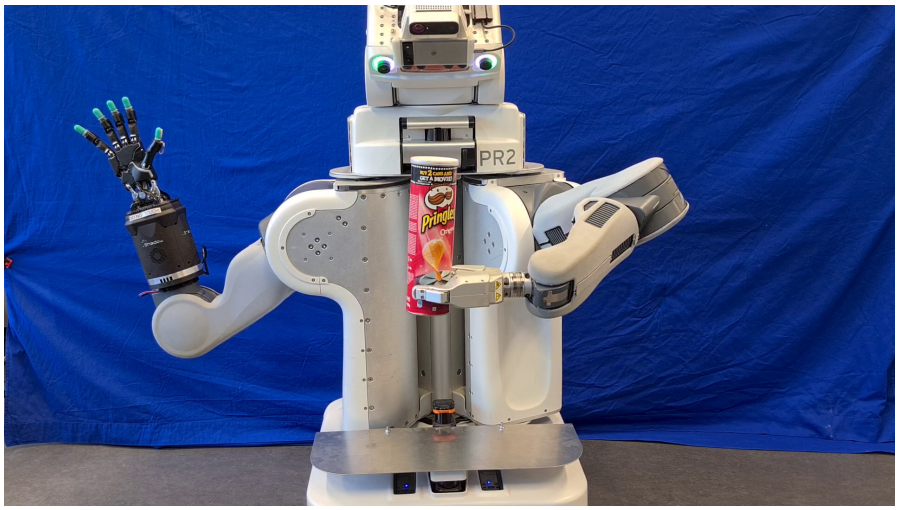


Figure 4.4.: Fixed pose to capture a point cloud of the handover object.

Point Cloud Filtering:

This pipeline uses the point cloud perceived by the Azure Kinect camera to choose the grasp point relative to the object in the gripper. The left arm moves into a predetermined pose to perceive the object. Figure 4.4 shows this pose. Besides having each object's height and width visible, it also ensures that the top of the object is visible in the point cloud for

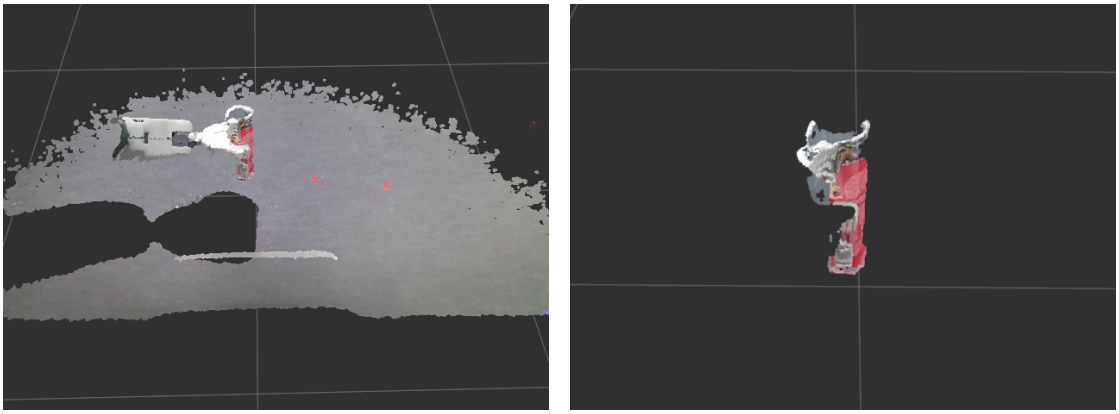


Figure 4.5.: On the left, an unfiltered point cloud received by the camera. On the right is the resulting point cloud after both filters are applied.

all objects used. The grasp selection process requires a reading of the depth of the object. While possibly problematic for other objects with a larger height or more complex shapes, this process works well for the chosen objects. A more refined strategy to generate the needed depth information for the object would require a full object exploration strategy and further point cloud refining, which goes above the scope of this thesis. For further discussion of the resulting issues from this simplified perception approach, refer to section 6.1.

Once it perceives the point cloud, the next step in the pipeline is to apply a crop-box filter above the gripper. This box spans an area above the gripper. Its dimensions extend 30 cm above the gripper and 10 cm along each direction of the x- and y-axes. Consequently, it ignores any points below the gripper, so the part of the object that the robot should grasp needs to extend above it. Filtering the point cloud this way in the first step ensures the background is filtered out, reducing the size of the point cloud significantly.

Finally, the remaining point cloud still contains parts of the gripper. These artifacts get filtered out by the already existing *robot_body_filter* package [49]. The resulting point cloud now only contains the object. Figure 4.5 shows an example of this two-step filter process.

Grasp Pose Selection:

The next step first transforms the point cloud correspondingly to a fixed gripper pose, allowing the selection of a grasp pose for the hand. This pose aligns the point cloud with the global coordinate frame. In this alignment, the z-axis corresponds to the height of the point cloud, while the x- and y-axes correspond to the depth and width.

The resulting point cloud then determines where to grasp the object with the Shadow hand. Currently, the pipeline allows two possible grasp modes at the start: top or side grasp. This choice determines the grasp pose. As it is only the starting hand pose for the later grasping process, the grasp pose only describes the pose of the *rh_manipulator* frame, which is attached to the Shadow hand, relative to the *l_gripper_tool_frame* of the gripper. Figure 4.6 shows both frames at their respective manipulator.

Now, the pipeline must calculate the pose according to the point cloud and grasp type.

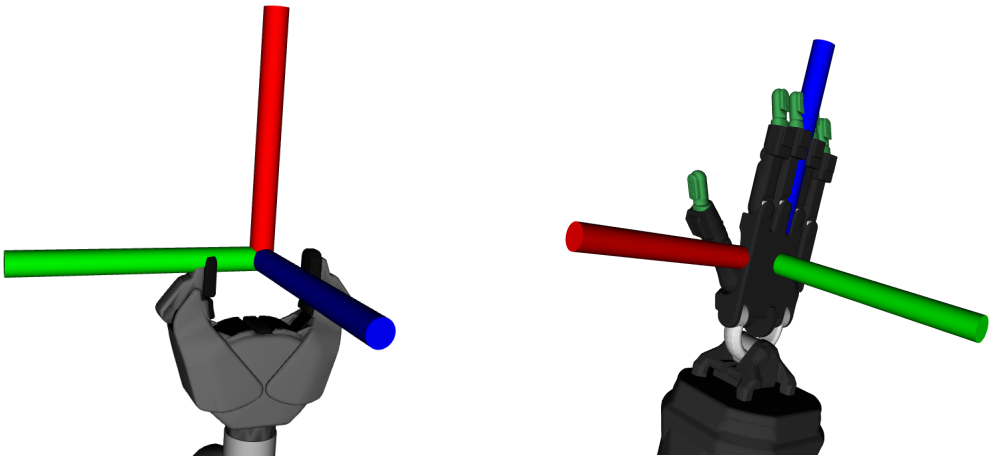


Figure 4.6.: The frames for defining the grasp pose for both manipulators. Left is the end-effector *l_gripper_tool_frame* for the left arm with the gripper. On the right is the *rh_manipulator* frame for the right arm with the Shadow hand.

Both types set fixed orientations of the hand relative to the gripper. Figure 4.7 shows examples of the orientation of both types. The flexible part of the grasp pose selection is the position. It also has static offsets, which apply to a position determined by the point cloud. For the top grasp, this flexible position has its z-value determined by the highest point of the point cloud. On the other hand, its x- and y-coordinates equal half the distance between the minimum and maximum values of the point cloud along these axes, respectively.

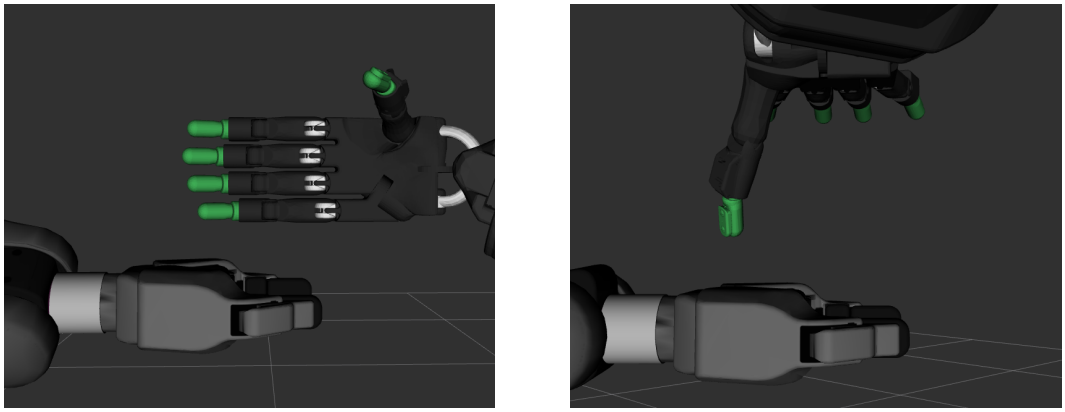


Figure 4.7.: Both available grasp types for the handover pipeline. The left image shows the default configuration for the side grasp, while the right one shows the configuration for the top grasp.

The hand has a fixed z-coordinate relative to the gripper for the side grasp. Further, the y-coordinate of the position for the side grasp follows the same method as for the top grasp. However, the x-coordinate equals the maximum x-coordinate in the point cloud.

As mentioned, static offsets then apply to this flexible position to compensate that the *rh_manipulator* frame does not align with the desired grasp point of the hand. Therefore, these offsets are the same for every object. The pipeline then passes the chosen grasp pose to the next part.

4.2.2. Handover Pose Sampling

This thesis follows the approach by Pavlichenko et al. [42] to decide where to perform the handover. They sample possible poses and assign a cost to every corresponding joint configuration. The pose sampling follows a predetermined grid in front of the robot. There are two possible sampling strategies available in this thesis. The first one is to sample from a subset of predetermined low-cost poses. Second is the option to sample random poses from the grid. Each pose results from a transform applied to the grasp pose generated by the previous module.

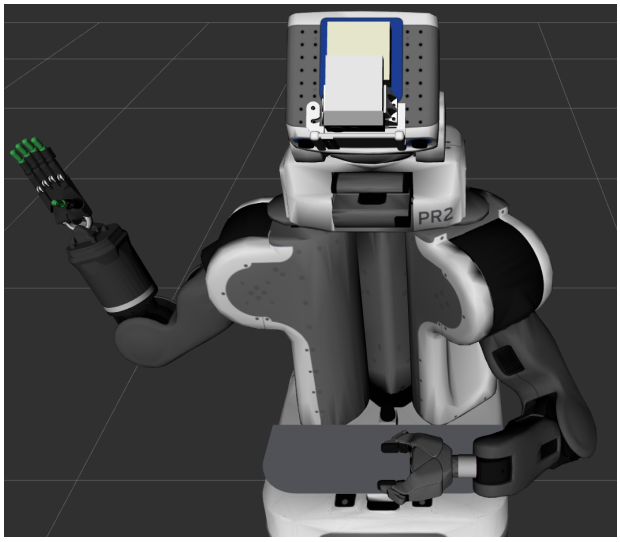


Figure 4.8.: Fixed starting pose for the handover pose sampling process.

The chosen transformations are applied as follows. Figure 4.8 shows the starting pose from where to apply the transformations. As described in the previous module, the gripper has a fixed pose, while the object's point cloud determines the hand's pose, which is specified relative to the gripper frame. Rotating both manipulators uses a new virtual handover frame. This frame is between the *l_gripper_tool_frame* and the *rh_manipulator* frame and has the same orientation as the global frame. For an example of the pose of this frame, refer to Figure 4.9. The rotational part of the transformation is then applied to both manipulators relative to this frame to ensure their relative orientation is maintained, keeping the previously chosen grasp pose. Afterward, the pipeline applies the translational part to both manipulators relative to the handover frame. Consequently, the pose of the handover frame can describe the poses of both end-effectors during the handover process. Therefore, this thesis references this pose further as the handover pose.

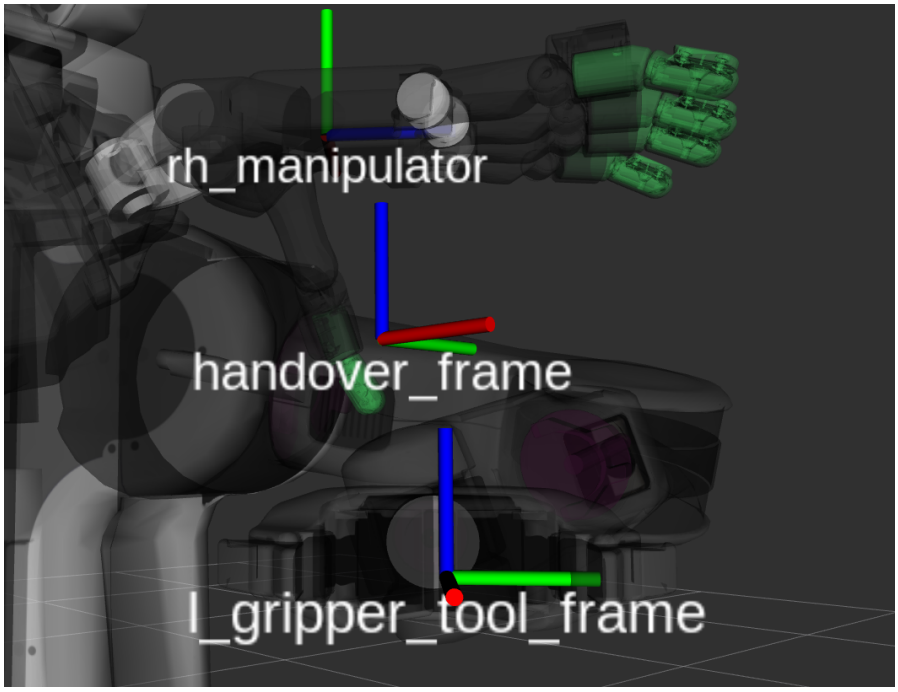


Figure 4.9.: The handover frame is a virtual frame between the two end-effector frames.

While this process describes how the sampling works for random handover poses, sampling from a subset of possible transforms requires a previously offline computed workspace analysis. Further description of this process follows in section 5.1.

During sampling, an IK solver calculates a corresponding joint configuration for each pose resulting from one of the sampled transformations. As mentioned, the IK solver used is BioIK [46]. First, it generates the configuration for the hand. Since the hand is more limited in its workspace due to the lower joint range in the wrist joints compared to the gripper, it is more likely not to find a valid configuration. The solver views configurations as invalid if they would result in a self-collision. If it finds no valid configuration, the sampled transformation is deemed invalid without generating the joint configuration for the left arm.

A valid configuration from BioIK needs further testing, as the solver allows approximate solutions. These approximate solutions enable configurations with a negligible distance to the correct pose, allowing the consideration of more transformations. However, this also means the solver will output an invalid solution when no valid solution exists. Consequently, the pipeline uses FK for the generated joint configuration to compute the resulting end-effector pose and test for the difference to the goal pose. Should the distance between the resulting pose and the target pose be over 1 cm, the process also regards this transformation as invalid. If the test is successful, the process repeats with the joint configuration for the left arm.

Next, the pipeline applies a cost function to each generated joint configuration to pick a transformation from all the sampled ones. It chooses the transformation with the lowest cost. The cost function follows the design by Pavlichenko et al. [42]. The idea of the cost

function is to reward configurations with their joint values far away from their corresponding joint limits. First, it calculates this distance using

$$\delta(\theta) = min(|\theta_{upper} - \theta|, |\theta - \theta_{lower}|), \tag{4.1}$$

where θ is a vector of all current joint values and θ_{upper} and θ_{lower} are each respective joint's upper and lower joint limits. The cost function is then as follows

$$c(\theta) = \frac{1}{|\delta(\theta)|} \sum_{i=1}^{|\delta(\theta)|} \frac{1}{\epsilon_i^2}(\delta(\theta_i))^2 - \frac{2}{\epsilon_i}\delta(\theta_i) + 1. \tag{4.2}$$

Here, ϵ_i is the maximum possible distance for joint i to its corresponding limits. While Pavlichenko et al. might have chosen a fixed ϵ value for all joints, the differences in the joint limits for the setup in this thesis make selecting individual values for each joint mandatory. The resulting cost is a value in the interval $[0, 1]$, where 0 would be the best and 1 the worst cost. Figure 4.10 shows a plot of the cost function. As it shows, the emphasis of the cost function is to have the joint values away from their limits, having a higher decrease in cost toward the limits in contrast to values already close to the maximum distance. Since the configurations of both arms need to be applied simultaneously, this process concatenates the joint values of both arms before the evaluation through the cost function.

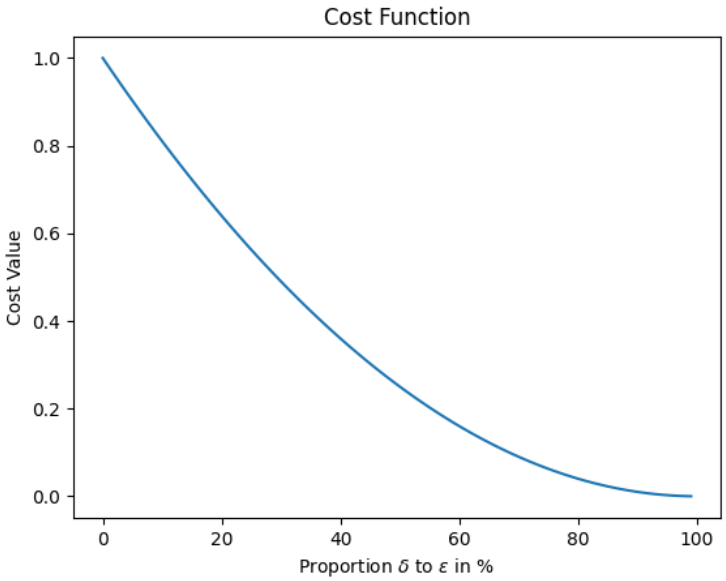


Figure 4.10.: Plot of the cost function used to evaluate the generated joint configurations for the values of one joint.

These configurations allow more flexibility for movement around the generated joint values. This characteristic is helpful for three aspects of this pipeline. First, it makes it more likely

for the gripper to have the flexibility to retract from the grasped object once the handover finishes. Configurations with a higher cost would have more likely scenarios where the gripper could only move away from the handover by moving through the object or pushing it away. This movement could push the object out of the hand's grasp or shift it so the grasp is less stable.

The second use of this flexibility is to ensure that the grasp test with the Shadow hand executes. Further description of this test follows in subsection 4.2.4. However, it involves a slight cartesian movement of the hand. With joints close to their limits, this movement might be impossible, resulting in no way to evaluate the grasp in training and normal execution. The final benefit of having more flexibility with the joint configuration is the higher likelihood for the approach of the hand to work, as later described in this section. As this approach is not tested for during the sampling, having the final configuration far from their joint limits makes it more likely to find a close configuration for the approach waypoint.

The sampling procedure stops if it has tested every configuration or the cost for one sampled transformation is below a threshold. Refer to section 5.1 for more information about this threshold. The sampling process returns the valid transformation with the lowest score if it evaluates all configurations without one passing the threshold. The robot then moves the gripper holding the object into the pose specified by the best transformation applied to the starting pose. Afterward, it moves the hand to its corresponding pose. First, the pipeline generates an additional hand configuration for a pose before the desired pose to help increase the likelihood of the hand moving to its sample configuration without colliding with the object. This pose is set at a fixed distance along a predefined axis for each grasp type. As no previous tests ensure this pose is reachable, the process skips this generated waypoint if it finds no joint configuration. Otherwise, the hand moves into this configuration and then to its sampled configuration. Then, the pipeline continues with the next step.

4.2.3. Grasping

This pipeline allows for two options to perform the grasp for the handover. All options use the same joints to perform the grasp and the same initial configurations. The Shadow hand joints used for the closing motion are the second and third joints on the four fingers and the thumb joints. The thumb starts by having its fourth joint set to 1.13 in radians to start in a pose that encourages moving the thumb parallel to the fingers. Depending on the grasp type, a top or side grasp, the fourth joints of the four fingers are set to either a slightly spread-out configuration or zero, resulting in a parallel setup. These initial values result in one of the two starting configurations shown in Figure 4.7. These initial configurations ensure a possible closing motion for all grasping options.

Both closing procedures also employ the same principles for detecting when the fingers have made contact with the object. This contact detection uses the tendon force sensors for each joint motor of the hand. Specifically, it uses the values of joints 2 and 3 for the parallel fingers and joint 5 of the thumb since the joints move the most during a closing motion, further referred to as the closing joints. It records these values before the grasping motion when the hand is in its initial grasping configuration at its grasping pose. Each contact check now compares the current value of a joint to these initial values. It perceives a contact if the

difference is above a set threshold. When one finger makes contact, it ignores its assigned values in further motions until all fingers have made contact. The effort difference for both joints of one finger must be above the threshold for the finger to have made contact.

The first closing option performs the grasping motion by adding a constant value to the joints at each timestep. Each joint has a fixed target value and a fixed number of steps for the closing motion to determine this constant value. The difference between the starting configuration for each joint and its target value is then divided by the number of allowed steps, giving the change applied to each joint during each step. After each step, the module checks all fingers for contact with the object. This option needs manual parameters set by a human with prior knowledge, acting as a baseline for further improvements.

The second option uses a neural network instead of human-designed fixed values to control the hand. It controls the hand using the concept of hand synergies explained previously in section 2.4. The network generates the weights for the first three synergies of the Shadow hand, which determine the joint changes at each timestep. However, unlike the second option, the neural network generates new weights at every step to determine how the joint values change. Later, section 4.3 explains this network's architecture and training process further.

Adding together the weighted synergies results in a vector with a value for each joint. These values are the changes for each joint and get added to the current value of all joints, resulting in the following joint configuration for the hand. As this can lead to erratic joint movements, especially during training, the pipeline limits the maximum change of each joint. If the maximum change of any joint would result in a change above 10°, it normalizes the change to 10°. The same normalization applies to all other joints to keep their relative movements. Adapting the grasp during execution allows reacting to sudden changes, such as object movement, during this process.

Once all fingers have made contact with the object using either closing option, the next module tests the grasp.

4.2.4. Grasp Testing

This pipeline utilizes the force/torque sensor in the wrist of the left arm to test an executed grasp. Since the force sensor is always in the same orientation relative to the gripper, its y-axis force measurement is perpendicular to the gripper's two fingers, as seen in Figure 4.11. Once grasping the object is finished, this module records the force values along this axis. Moving the hand about 5 mm along this axis while grasping the object allows for comparing the resulting force with the previously recorded values. The grasp is estimated to be successful if the difference is above a threshold.

This testing relies on Equation 2.3. The fingers apply a normal force against the object through the previous grasp closing. By pulling up the object, the tester measures the resulting tangential force. If it is large enough, it indicates the grasp to have enough quality to avoid slippage.

Once the test estimates the grasp to enact sufficient force, the gripper opens, releasing the object to be held by just the hand, and retracts. The hand can then move the object towards the goal pose.

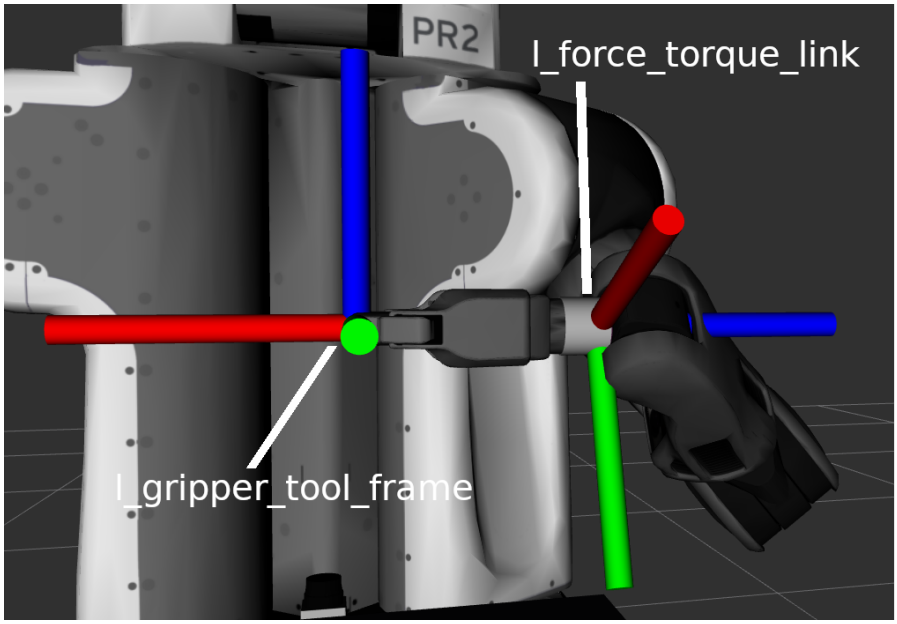


Figure 4.11.: The wrist-mounted force sensor relative to the gripper end-effector frame.

4.3. Training

As mentioned in subsection 4.2.3, the training only aims to provide a model for closing the Shadow hand for an object grasp during the handover process. All of the training process only uses the real-world robot. This thesis takes inspiration from the work of Liang et al. [22] to allow this training process to work. It relies on their collected hand synergies. The first section of this chapter describes the RL architecture for this training. Afterward, the second section explains the execution of the training process on the real-world robot. For further evaluation of this training procedure, refer to section 5.2.

4.3.1. Architecture

This part explains the structures used for the RL process. It starts with an overview of the RL framework, algorithm, and hyperparameters. Afterward, it explains the definitions of the action and state space. Finally, it covers how the RL process calculates the reward for the training. As this training structure is the basis for the training and evaluation in section 5.2, its design focuses on working with the three objects used later.

Parameters:

The training relies on the framework *stable-baselines3* [50] for training the neural network. It provides implementations for a variety of online reinforcement learning algorithms. As mentioned, this thesis uses SAC as its RL algorithm. As an off-policy model, it uses a replay buffer to be more efficient with the collected samples, which is beneficial for training on a real robot. The hyperparameters for the training procedures are mostly left unchanged, as SAC

is supposedly resistant to suboptimal hyperparameters, and an exhaustive hyperparameter optimization process is unfeasible on a real-world robot. One exception to this is the entropy coefficient for SAC. As it is one experiment, subsection 5.2.1 describes this selection process since this parameter needs to be set for each task individually. The default architecture for the MLPPolicy of the SAC algorithm in stable-baselines3 has it set for both actor and critic to be two fully connected 256 hidden layers between the input and output layers. This layout is also the network architecture for this thesis.

Action Space:

Two essential factors for the training time of a RL model are its action and state space. In robotics tasks, they often have to be continuous, relating to sensor input or joint values. This continuous nature is also the case in this thesis. Correspondingly, the amount of possible states and actions the agent has to explore is already enormous. However, this property is a requirement for these tasks. The other alternative to reduce the number of possible actions and states, and, therefore, reducing the required training time, is reducing the dimensionality for both. Using hand synergies, as explained in section 2.4, allows this reduction for the action space. Liang et al. collected these already for a Shadow hand. As their Shadow hand is a left hand, using them requires transferring the values to the corresponding joints on the right hand used in this thesis. Since this hand has BioTac sensors, it also misses the first joints on each finger. Correspondingly, the pipeline ignores the values for these joints when using the hand synergies. Using only the first three hand synergies to control the hand enables a reduction to three dimensions. These three values are the network's output. They are values normalized to the interval $[-1, 1]$.

State Space:

The state space, and, therefore, the network input, contains three parts. The first one also uses hand synergies, as does the action space. However, unlike previously, the joint configuration gets projected onto the first three hand synergies. Through this projection, one can compute the weights for the synergies that would generate this configuration most closely. While this process loses some information about the exact joint configuration, the network can learn to compensate for it. This encoding provides the network with information about the current state of the joints in a significantly reduced dimensionality, the same as for the action space. Specifically, it reduces the information to three values as the first part of the state space.

Next are nine values corresponding to the effort difference for all nine closing joints. These effort values are the same ones used for the contact check with the object. The Shadow hand provides these values through the force sensors at the tendons that control the finger motors. These sensors are very susceptible to environmental changes. Also, the simple movement of the hand from one pose to another can significantly change these values. Since values that change independent of the task required from the network would be unsuitable inputs, the state space instead contains the difference in these values. This process is similar to the one for contact detection explained in subsection 4.2.3.

Finally, a three-dimensional one-hot encoding of the current target object completes the state space. This one-hot encoding is a vector where one value is 1 while the others remain

0. Each object corresponds to one of these values. While still requiring human input for this value, it is the only information used on an object and does not require any information about its shape. Further, while adding three more values to the dimensionality, these values are static over one training episode, making it easier for the network to learn the impact of these values. This value needs to be set for new objects to the object the network was trained with that most closely matches the new object. Consequently, this results in a state space of 15 values normalized to $[-1, 1]$.

Such a normalization is standard in RL approaches to improve the training results. Since the one-hot encoding is already in the normalized range, only the synergy encodings and the effort values need to be normalized. For the effort values, the normalization process caps them at double the threshold value that would detect an object contact, both positive and negative. Then, they get divided by this cap, resulting in the desired interval. The encodings are more difficult to normalize, as no guaranteed limits exist for any component. For the first three synergies, dividing the values by three has experimentally shown to let the values always fall into the desired range.

Reward:

The reward function for the network takes inspiration from the one designed by Liang et al. [22]. It defines two reward cases, which are dependent on the timestep of the training. Each timestep is one change of the hand joint configuration through the network. One episode includes all timesteps until one grasping attempt finishes. Consequently, the reward function defines a reward as

$$r_t = \begin{cases} r_b + r_{con}, & \text{if } t = T_{final} \\ r_c, & \text{otherwise} \end{cases} \tag{4.3}$$

with r_t the reward at step t during on training episode and T_{final} as the episode length. Therefore, the agent gets one reward type at the end of an episode, while it gets another at the timesteps during the episode. The closing reward r_c depends on how much the closing joints change towards their closing direction. It equals the sum of the change in the joint value for each closing joint in radians, specifically

$$r_c = \sum_{i=1}^{17} q_i(t) - q_i(t-1) \tag{4.4}$$

where $q_i(t)$ is the value of joint i at timestep t for all 17 joints of the Shadow hand. All closing joints for the Shadow hand have their highest value when the hand is closed. Correspondingly, this reward is positive if the overall change moves the joints further toward a closing pose and negative if the hand moves into an open pose. For the other rewards, r_b is a binary reward whether the grasp testing at the end of an episode deems the grasp stable. It has a value of 1 or -1, depending on whether the test succeeded or failed, specifically

$$r_b = \begin{cases} 1, & \text{if grasp successful} \\ -1, & \text{otherwise} \end{cases} \tag{4.5}$$

In addition to this reward, the end of an episode also provides a contact reward r_{con}. It equals the number of all closing joints above the threshold for contact detection multiplied

by a weight of 0.05, defined by

$$r_{con} = 0.05 * \sum_{i=1}^{9} con(i) \tag{4.6}$$

$$con(i) = \begin{cases} 1, & \text{if closing joint i made contact} \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

Consequently, the contact reward is a value in the interval $[0, 0.45]$. This additional reward encourages grasps with more finger contacts, which are assumed to be more stable. While it can improve the reward for a failed grasp, its maximum value ensures that the combined reward remains negative.

4.3.2. Procedure

When starting training, the robot moves into the initial configuration by executing the first step of the pipeline. Next, a human operator must specify the chosen object and the desired grasp type. Afterward, the pipeline continues until the start of the grasp phase. This process results in both gripper and hand being in the correct configuration to perform grasps on the object. Now, the algorithm performs several grasp attempts on the object. These attempts repeat until it has executed around 1000 steps on the current object, ending this iteration. More specifically, since the episode length can vary but is limited to 50 steps as a maximum, the training for the object stops when the first episode, which brings the number of executed steps over 1000, finishes. Then, the robot moves into its initial configuration again, and a human operator can exchange the object for the next one. This process repeats for 12000 steps.

Each grasp attempt consists of up to 50 grasp steps. Each step calls the previously defined network to generate its three output values. These values produce the following joint configuration for the hand, as explained in subsection 4.2.3. The resulting joint configuration is then sent directly to the controller of the Shadow hand for execution. Unlike other movements, this skips the collision-aware planning for the hand to speed up the training. However, this speedup entails the cost of limiting the hand joints during execution to prevent the hand from potentially damaging itself. Specifically, the training does not use the fourth joints of each parallel finger, as they could produce collisions between the fingers. After each step, the reward function described in subsection 4.3.1 generates the appropriate reward for this hand movement.

The network output values are further limited to guide the synergy-controlled hand into a closing motion. As seen in Figure 4.12, the first two synergies have a distinct closing direction through negative weights. Therefore, before the pipeline uses the network's first two output values, it normalizes them to the range $[-1, 0]$ to enable only a closing motion. On the other side, the third component has no visible closing direction, so the original output value of the network remains.

One grasping attempt stops once it has run for 50 steps or all fingers have made contact with the object. This limit ensures that the grasping process does not go on indefinitely

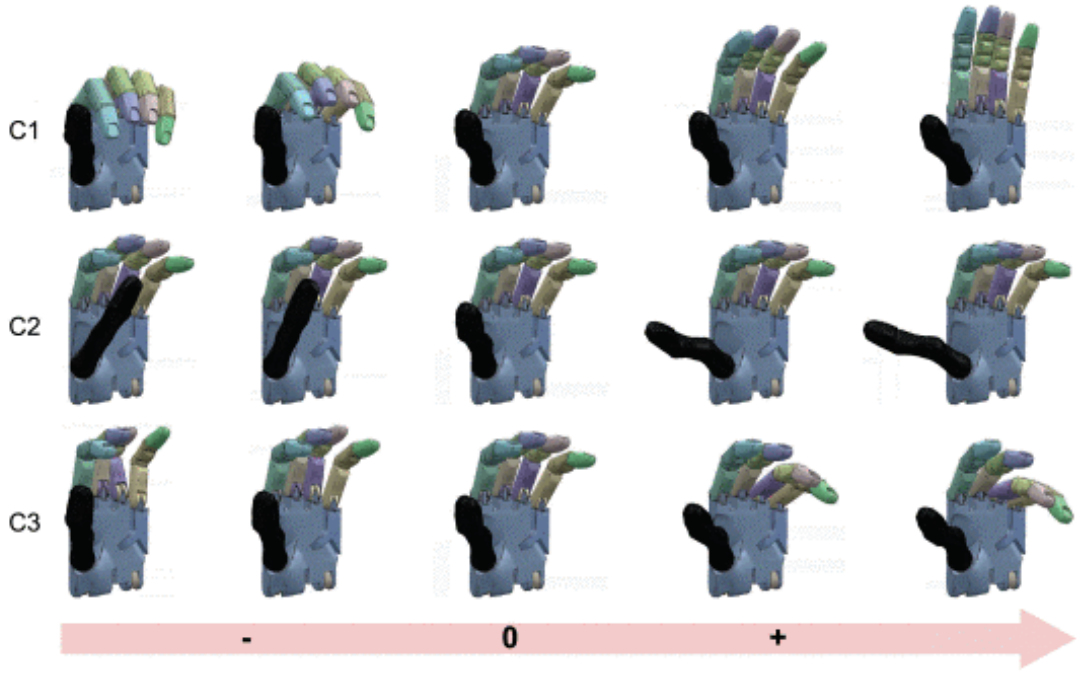


Figure 4.12.: The first three synergies for the shadow hand and their respective movements for positive or negative weights. [22]

while guaranteeing that each model attempts a minimum of 20 grasps during each 1000-step iteration. At the end of each attempt, the reward function generates the final reward based on the grasp testing described in subsection 4.2.4.

This training process allows for semi-self-supervised execution by the robot. Theoretically, a human operator must only replace the training object every 1000 steps and specify the grasp type. During each iteration, the gripper never releases the object, so it does not need to be replaced by a human after each attempt. Further, the grasp testing strategy for reward generation allows the process to reward the agent without necessary input by a human. In practice, potential hardware failures or possible movements of the object in the gripper during an iteration still require an operator always to be present. However, this reduced required oversight is necessary for the process of real-world RL to be feasible.

Training on a real-world robot also needs to be able to handle training interruptions. These can come, for example, from hardware failures or lost sensor information due to latency issues. Stable-baselines3 allows the saving of model checkpoints after a set number of steps. For this thesis, the training saves checkpoints after every 100 steps. Whenever an issue interrupts a training process, it can continue after the problem is solved by loading the last valid checkpoint.

5. Experiments

Two experiments comprise the pipeline's evaluation. First is the workspace analysis mentioned for the PR2 configuration. Second is the evaluation of different trained models for the grasping part of the pipeline. This chapter presents both experiments in detail and their results.

5.1. Workspace Analysis

The analysis of the PR2's available workspace for bimanual manipulation consists of sampling different configurations over a grid in front of the robot. It performs this analysis offline. This grid contains positions 6cm apart along all three axes. For each position, the analysis also tests rotations. These rotations consist of all possible combinations of -90° to 90° in 30° steps for each axis, resulting in 343 rotations tested for each position, as seen in Figure 5.1. The rotations apply to the hand and the gripper around the handover frame to keep their poses relative to each other the same.

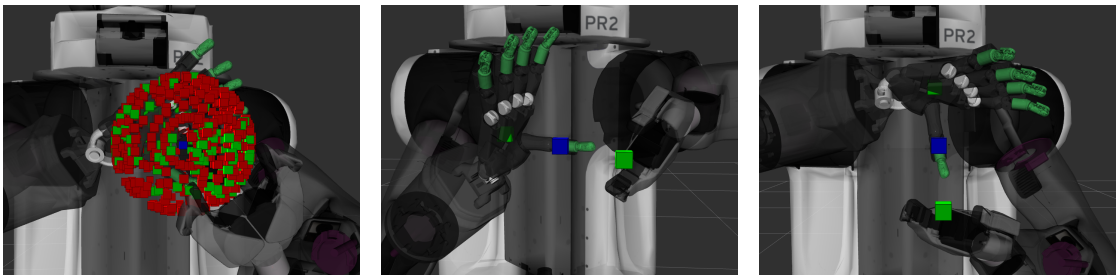


Figure 5.1.: Example of sampled rotations for one handover pose (blue). Green shows a configuration where both manipulators produced a valid solution, and red indicates invalid ones. All sampled poses for both hand and gripper are on the left. Each rotation for the hand has a corresponding rotation for the gripper opposite of the handover pose and vice versa. The middle and right images show examples of these corresponding rotations.

This analysis uses three metrics for every position to evaluate its performance. First is the number of rotations for the position that results in a valid solution. This metric explains how flexible a position is for the considered configuration. More possible solutions allow a higher likelihood of finding a solution when including this position in the online sampling. Next is the average cost of all valid solutions for one position. On its own, this metric favors position with few low-cost solutions. Combined with the previous metric, however, it could allow one to find a position with high flexibility that generally produces low-cost solutions. Finally, the last

metric is the best solution cost for one position. This metric gives an insight into a possible threshold for an early stop during the online sampling. Further, combined with the previous two metrics, it may provide a region of positions that can generate multiple, low-cost, valid solutions with the potential for early stopping to speed up the online sampling process.

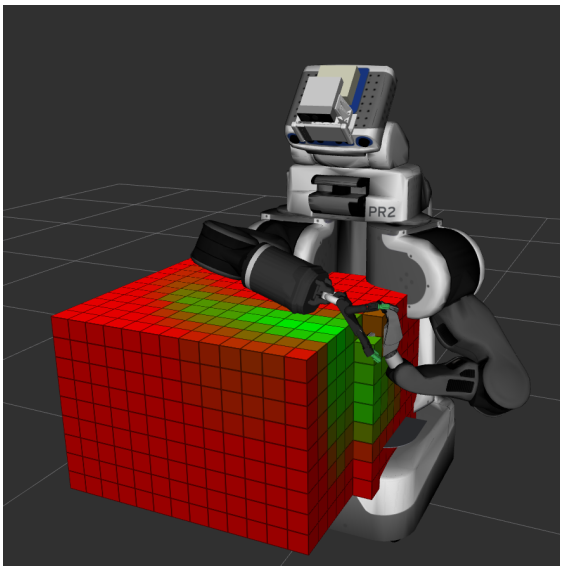


Figure 5.2.: An example of creating the grid-based workspace analysis for the side grasp.

The grid is dynamically grown in front of the robot to ensure the analysis tests the entire workspace of the robot, independent of possibly faulty human intuition. This growth starts with a single position in front of the robot. The analysis then calculates the previously defined metrics of all transformations related to this position, so all combinations of the previously specified rotations with this position. Only the number of valid solutions for this position is relevant to growing the grid. Along all axes, in both the positive and negative directions, a check is made to determine if any border positions still have valid solutions. If that is the case, the size of the previously checked block extends by one step in this direction. Once it has checked every side for expansion, the analysis calculates the performance for all added positions again. Figure 5.2 shows an example of creating the grid through this process. This process repeats until no further expansion is required, resulting in a wholly grown area for the workspace analysis. The number of tested transforms will likely change depending on the tested configuration.

This experiment aims to apply the previously defined grid search and metrics to both available grasp types and slight deviations from one of these. As the analysis requires the same grasp pose for each checked transformation, it works with fixed values instead of getting the hand pose relative to the gripper from a point cloud. For each configuration, it generates the best sampling regions according to the metrics and investigates the change between them.

5.1.1. Side Grasp

The first workspace analysis regards the side grasp. It applies each of the three metrics to the resulting workspace. The grid of the analysis grew according to the previous method until no position on any border of the grid provided a valid solution. This procedure resulted in a grid of 5670 positions.

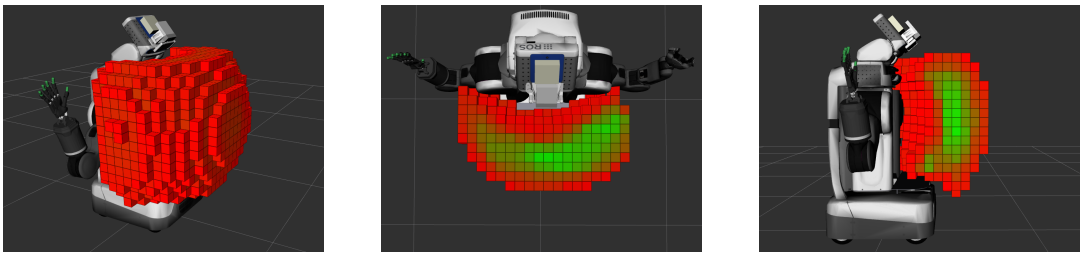
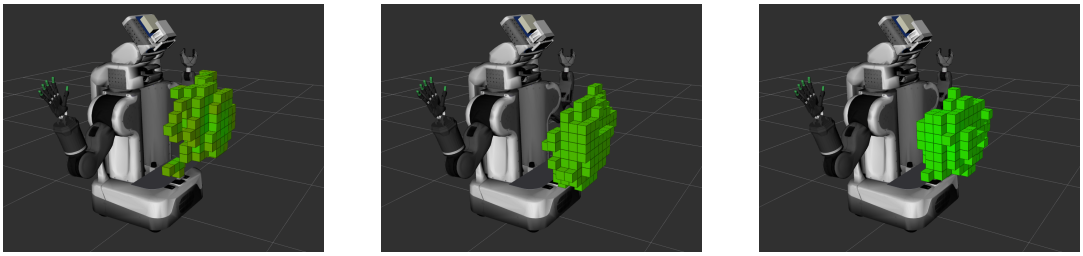


Figure 5.3.: The left image shows all positions with at least one valid solution for the side grasp. In the middle and right images, a cut of half of this distribution highlights the inner structure along the xy- and xz-plane, respectively.

These valid solutions form an arc as shown in Figure 5.3. Each visualization and any other workspace images in this section have additional views in section A.1. This part is central in front of the robot. A spherical pattern is typical for manipulator workspaces, so having a similar curved shape for the combined workspace is a plausible result. The arc's cutout is close to the robot's body, as collisions with said body make solutions invalid. Also, the arc ends at each side with a manipulator where the range of the other manipulator ends. This end is sudden at the left arm's side, indicating a high manipulability even at the edge of the reachable space for the right arm.

So far, this is the expected behavior of the workspace. However, the inner part of this workspace is more interesting. While it has the expected structure of an inner core with the most valid solutions, while the number of solutions shrinks towards the edges of the arc, this core is not central in front of the robot. Instead, it leans towards the left arm with the two-finger gripper. One explanation could be limited flexibility in the wrist joints of the hand.



(a) Number Solutions (b) Average Cost (c) Minimal Cost

Figure 5.4.: Best 5% of positions according to the three defined metrics of number of valid solutions, average cost, and minimal cost for the side grasp.

Consequently, reaching poses close to the arm, where the arm would need to be angled more, is more challenging. The gripper, on the other hand, has a more flexible wrist structure. This flexibility also allows it to reach poses where the left arm needs to be angled significantly more, explaining the observed core shift.

Like Zacharias et al. [15], the analysis also considers the top percentage of valid solutions in Figure 5.4. In this case, this is the best 5% of solution. As expected, it coincides with the inner core of all viable solutions. It also has the observed shift toward the left arm. The analysis also applies this process to the other metrics, namely average and minimal cost, as seen in the figure. These also correspond to the inner core with only slight variation in shape.

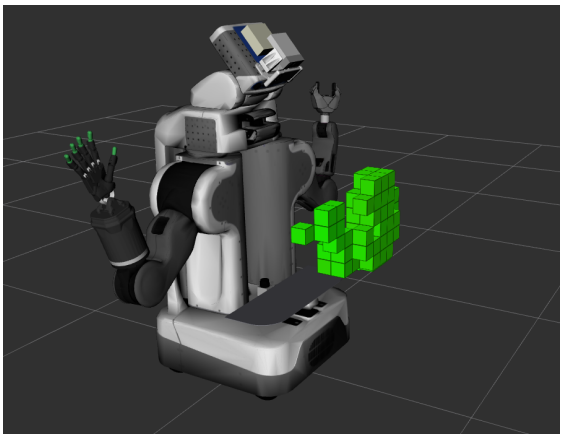


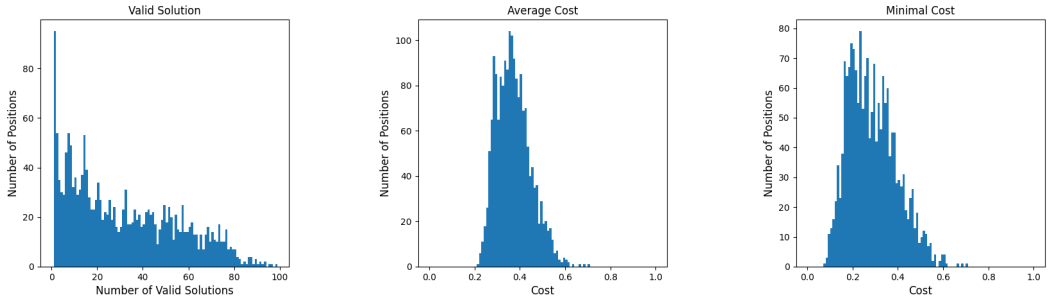
Figure 5.5.: The intersection of the best 5% of positions for each metric for the side grasp.

The final step computes the intersection of the best 5% of positions for each metric to filter out these differences. Refer to Figure 5.5 for the result. It follows the other three metrics in general shape but with reduced size due to the minor differences. According to the metrics, this region now contains several positions likely to produce multiple valid, low-cost handover poses. However, because the analysis only used one fixed pose for the hand relative to the gripper, the pipeline needs to check these handover poses online again, as the handover pose varies based on the point cloud of the object. The following section repeats this analysis for variations of the side grasp to investigate how this variation might influence the optimal sampling region.

However, to ensure the online chosen handover pose also has a low score, the sampling proceeds until a pose has a score below a threshold. The cutoff point for the best 5% of the minimal cost metric determines this threshold. This process set the threshold to 0.178 for the side grasp. Table 5.1 provides an overview of this and other cutoff values for this and every other analyzed grasp. Another part of the analysis viewed the distribution of all metrics and looked for anomalies. As Figure 5.6 shows, no metric has such an anomaly. Both cost metrics have a skewed normal distribution, while the number of valid solutions shrinks linearly. This shape is identical to the plots for each tested configuration in this chapter. Consequently, these plots are in Appendix A.

Table 5.1.: Overview of cutoff values for the different workspace analyses. Grid size equals the number of investigated positions for the corresponding workspace analysis of the specified grasp type. Cutoff values are the limiting values between the top 5% of positions according to that metric and the rest. For the number of valid solutions metric, the top 5% have equal or more than the given value, while the value needs to be equal or lower than the value for the other two metrics.

	Side Grasp	Side Grasp X-Shifted	Side Grasp Y-Shifted	Top Grasp
Grid size	5670	5940	5670	7392
Valid Solutions Cutoff	59	57	54	47
Average Cost Cutoff	0.291	0.289	0.292	0.298
Minimum Cost Cutoff	0.178	0.183	0.184	0.137



(a) Number Solutions (b) Average Cost (c) Minimal Cost

Figure 5.6.: The distribution of all valid solutions for the side grasp according to the three defined metrics of number of valid solutions, average cost, and minimal cost.

5.1.2. Shifted Side Grasp

As mentioned, this chapter repeats the analysis for slight variations of the side grasp. Specifically, it considers two variations: a shift along the positive x-axis along the handover frame and a shift along the negative y-axis. As the side grasp has a fixed value along the z-axis relative to the gripper, these are possible variations that can occur by changes in the point cloud for the different objects.

For the x-shift, the structure of all valid solutions is similar to the side grasp, as seen in Figure 5.7. As with the side grasp, additional views exist in section A.2. Interestingly, it does not have a clear shift along the x-axis, as might be expected. Instead, the grid extends by one layer of positions along the positive y-axis. This extension is likely the result of the

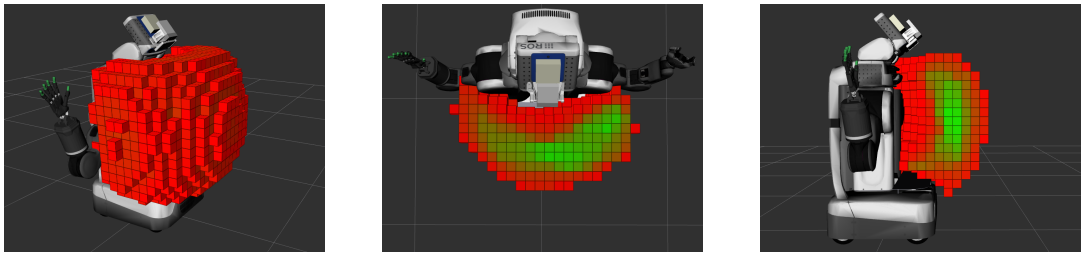
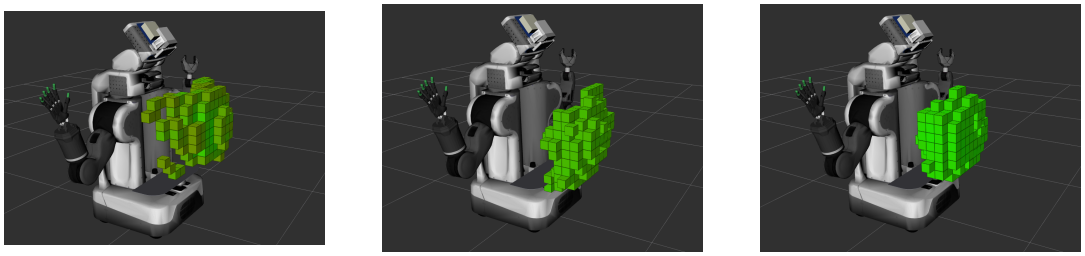


Figure 5.7.: The left image shows all positions with at least one valid solution for the x-shifted side grasp. In the middle and right images, a cut of half of this distribution highlights the inner structure along the xy- and xz-plane, respectively.



(a) Number Solutions (b) Average Cost (c) Minimal Cost

Figure 5.8.: Best 5% of positions according to the three defined metrics of number of valid solutions, average cost, and minimal cost for the x-shifted side grasp.

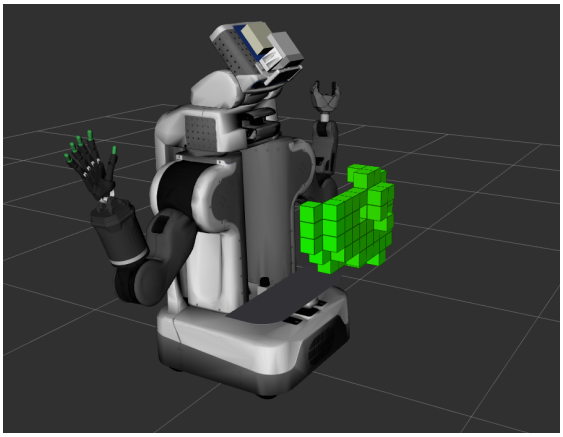


Figure 5.9.: The intersection of the best 5% of positions for each metric for the x-shifted side grasp.

greater distance between the hand and the object, enabling these positions through rotations that change this distance towards the y-axis. As previously observed with the side grasp, the manipulability is still high along this edge. Further, the inner structure is also similar.

Neither is a clear shift visible for the top 5% of positions according to each metric, shown

in Figure 5.8. This similarity is also visible for the cutoff value of each metric. While slightly worse regarding the number of valid solutions and the minimum cost metric, the values remain similar. Correspondingly, the intersection visible in Figure 5.9 also remains similar.

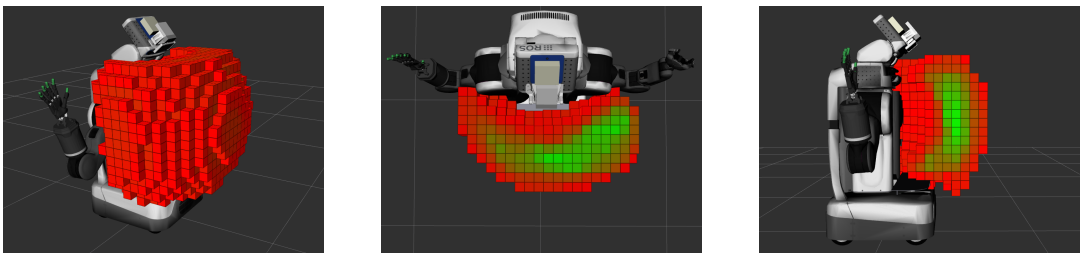
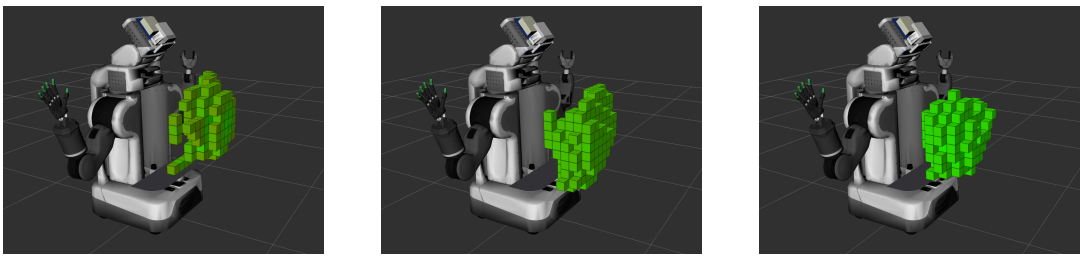


Figure 5.10.: The left image shows all positions with at least one valid solution for the y-shifted side grasp. In the middle and right images, a cut of half of this distribution highlights the inner structure along the xy- and xz-plane, respectively.



(a) Number Solutions (b) Average Cost (c) Minimal Cost

Figure 5.11.: Best 5% of positions according to the three defined metrics of number of valid solutions, average cost, and minimal cost for the y-shifted side grasp.

The y-shifted side grasp also produced a similar structure for all valid solutions, seen in Figure 5.10. As before, section A.3 has additional views for each visualization. The grid has the same extension towards the positive y-axis but one less layer along the negative y-axis. Correspondingly, the analysis covered the same number of positions as the original side grasp. Otherwise, the inner and outer structures remain similar to the previous two grasp types.

The metric analysis remains, therefore, very similar to the side grasp, as seen in Figure 5.11. As with the x-shift, no significant changes occurred. The intersection, depicted in Figure 5.12, also remains similar to the side grasp. Only minor changes happened for the cutoff values, each performing slightly worse than for the side grasp.

These minor changes for both shifts show that using the same sampled handover poses for one grasp type is possible even when variations during the online handover change the exact handover pose. As the cutoff value for the minimum cost climbed above 0.18 for the two shifted grasps, online sampling for side grasps will use 0.19 as its threshold.

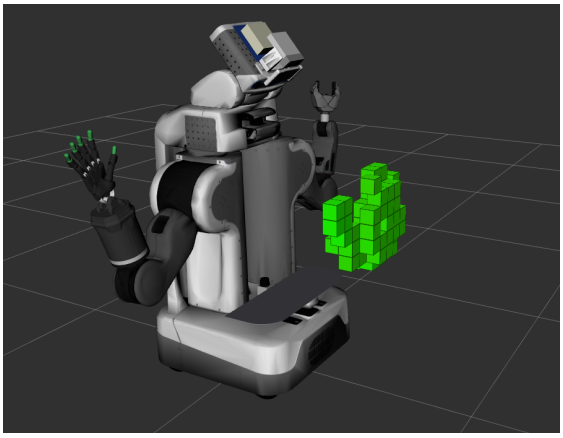


Figure 5.12.: The intersection of the best 5% of positions for each metric for the y-shifted side grasp.

5.1.3. Top Grasp

Finally, the workspace analysis also covers the top grasp. As seen in Figure 5.13, the shape of all valid solutions differs significantly compared to the side grasp. Additional views in section A.4 further highlight this. First, the grid size increased significantly to now consisting of 7392 positions. These positions come from three additional layers along the positive x-axis, two along the negative z-axis, and one along the negative y- and positive z-axis. However, it also lost two layers along the negative x-axis. Second, the still present inner core, where the positions with the most valid solutions lie, is more centered and no longer moved toward the left arm. Since the top grasp likely has more distance between the hand and the gripper, this observation adds to the previous theory about the shift for the side grasp. Additionally, it is further away from the robot's body.

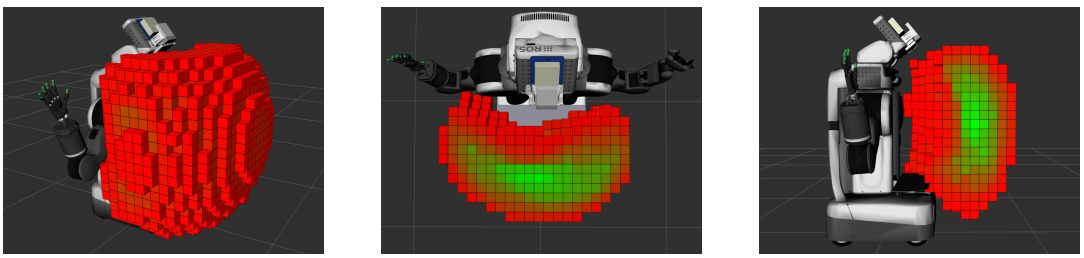
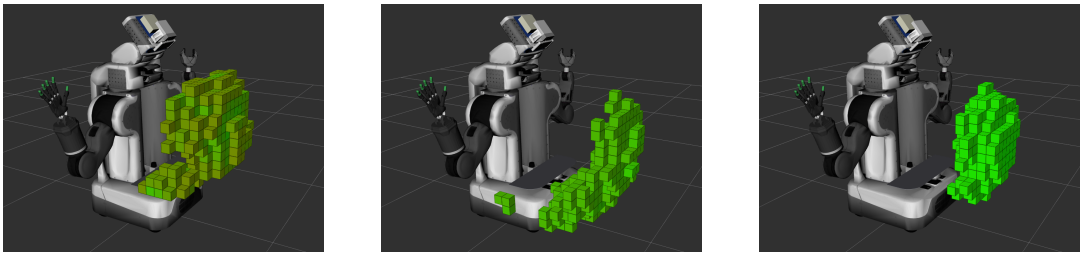


Figure 5.13.: The left image shows all positions with at least one valid solution for the top grasp. In the middle and right images, a cut of half of this distribution highlights the inner structure along the xy- and xz-plane, respectively.

The metric analysis for the top grasp also produced significant differences from the side grasp. As seen in Figure 5.14, every metric contains more positions in its top 5%, since the overall number of valid positions also increased. For the minimum cost, the top grasp has the slightest difference compared to the other metrics. It is still a centered, continuous core,



(a) Number Solutions (b) Average Cost (c) Minimal Cost

Figure 5.14.: Best 5% of positions according to the three defined metrics of number of valid solutions, average cost, and minimal cost for the top grasp.

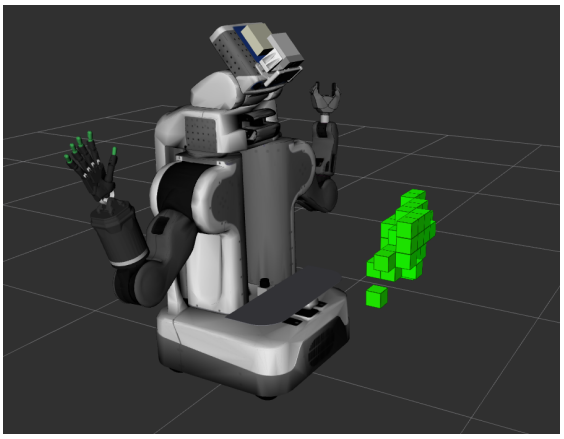


Figure 5.15.: The intersection of the best 5% of positions for each metric for the top grasp.

which extends further along the negative z-axis and starts further away from the robot, as was already observed to be the case for the entire structure. On the other hand, while still having a centered, continuous block, the number of valid solutions metric now also has a smaller, almost separate block towards the right arm. This block is not far from the central one but further towards the robot. It likely highlights a region where the right arm might reach easily, even with limited wrist flexibility, producing an optimal zone. Nonetheless, it highlights why such a workspace analysis is helpful for further understanding such a complex problem, as this behavior is counter-intuitive to human expectation.

Finally, the average cost metric produced the most different results. Instead of remaining a shifted block, it formed a relatively flat shape that follows an arc towards the negative z-axis. While still central, it also starts further away from the robot than the other two metrics for the top grasp. An explanation for this behavior would be that these positions only produced a low number of valid solutions, as they were not part of the number of valid solutions 5%, which had a relatively low cost. They are not part of the top 5% of minimal cost, so their cost is likely just above this threshold. However, according to the minimal cost metric, the best positions likely produced several other valid poses with higher costs. Since the average

cost metric only considers valid solutions, it prefers these positions. More surprisingly, this phenomenon did not occur for the side grasp.

Due to the significant differences between the metric results, the resulting intersection, seen in Figure 5.15, contains fewer positions than the side grasp. As expected from the grid shift along the x-axis, it is further away from the robot's body. Due to the increased number of positions in the grid, the cutoff value for the number of valid solutions is also lower. The average cost also has a worse threshold, increasing to almost 0.3. However, the minimum cost significantly decreased to 0.137. It indicates that the workspace for the top grasp allows for more optimal handover poses. Since the threshold varied for the side grasp with shifts, the threshold for an online sampling of top grasps will, therefore, be 0.15. Ultimately, the analysis provided several positions to sample from for online top grasps. However, it showed how different the workspace can be for varying grasp types. This difference highlights the requirement to perform such an analysis for any future additional grasp type.

5.2. Model Training

As mentioned, the entire RL training runs exclusively on the real-world robot system. The training procedure follows the outline explained in subsection 4.3.2. Training the models and comparing them with the baseline grasp closer uses the objects depicted in Figure 5.16. The chips can and bleach bottle are part of the YCB Object and Model Set [51]. However, the plastic cap is colored with white paint to prevent issues with the point cloud readings for the top of the chips can. The paper roll also has a paper cap on its top for the same reason.

Figure 5.16.: The three objects used for training and testing the grasping models and the handover baseline. They consist of a chips-can, a bleach bottle, and a paper roll from left to right. The left image shows the front view, while the right image depicts a top view.

5.2.1. Hyperparameter Selection

Since the training runs entirely on the real-world system, a standard hyperparameter optimization is unsuitable for this thesis. Such an optimization usually consists of a grid search

Table 5.2.: Configuration for all training scenarios during hyperparameter selection.

	Training Attempts					
Number	1	2	3	4	5	6
Steps	24000	12000	12000	12000	12000	12000
Entropy Coefficient	1.0, Adaptable	1.0, Adaptable	0.1, Adaptable	0.05, Constant	0, Constant	0.001, Constant
Input	Full	Full	Full	Constant	Full	Full
Poses	Variable	Fixed	Fixed	Fixed	Fixed	Variable
Reward	Original	Original	No Intermediate	No Intermediate, Timescaled	No Intermediate, Timescaled	No Intermediate, Timescaled
Training Time	5.0 h	1.5 h	1.4 h	1.7 h	2.6 h	3.5 h

over different hyperparameter combinations and training a new model with each combination. Such a strategy is unfeasible with the training time on a real-world robot. Fortunately, the original SAC paper [26] highlights the relative robustness of SAC to suboptimal hyperparameters. Unfortunately, they also mention one exception to this rule, the entropy coefficient or reward scale. The authors mention that this parameter needs tuning for each environment. Therefore, determining a suitable value for this coefficient requires some exploration, which the rest of this section explains. All these explorative training uses the side grasp setup explained in subsection 4.2.3.

All in all, tuning the hyperparameter required six training attempts. Refer to Table 5.2 for a list of configurations for each attempt. Details about these configurations and attempts

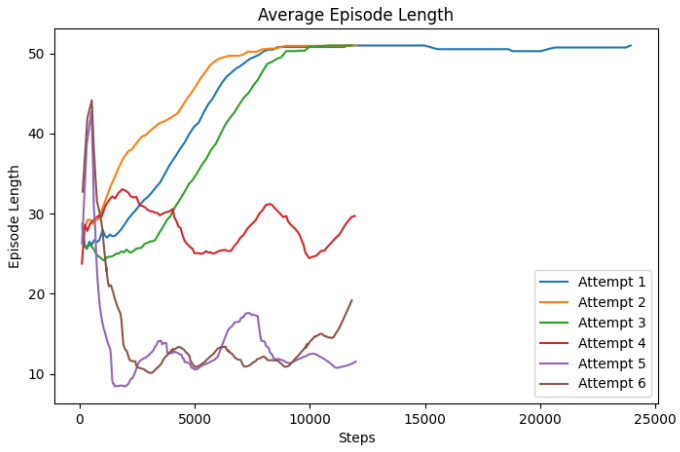


Figure 5.17.: Average episode length for all training attempts. The Table 5.2 shows the configurations for all attempts.

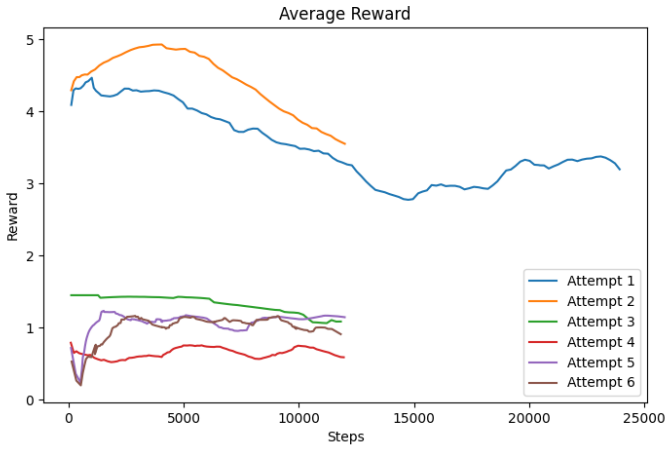


Figure 5.18.: Average reward per episode for all training attempts Table 5.2 shows the configurations for all attempts. The reward function changed over different training attempts, so higher values do not necessarily relate to better attempts.

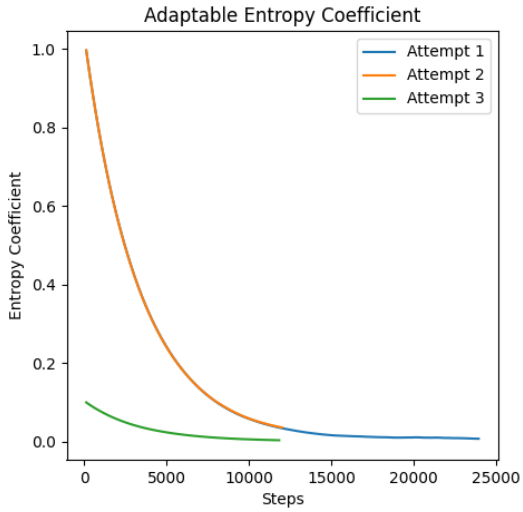


Figure 5.19.: Entropy coefficient for all training attempts with a variable one. The Table 5.2 shows the configurations for all attempts.

follow in the rest of this section. The first training attempt was to determine if the default settings provided by stable-baselines3 were already suitable for the scenario of this thesis. For the entropy coefficient, the default option is a method to adapt the coefficient during the learning process. Starting with a value of 1, it gets learned in parallel to the network parameters. However, this setting led to a deteriorating policy. As seen in Figure 5.17, the average episode length increases over time while the average reward decreases, visible

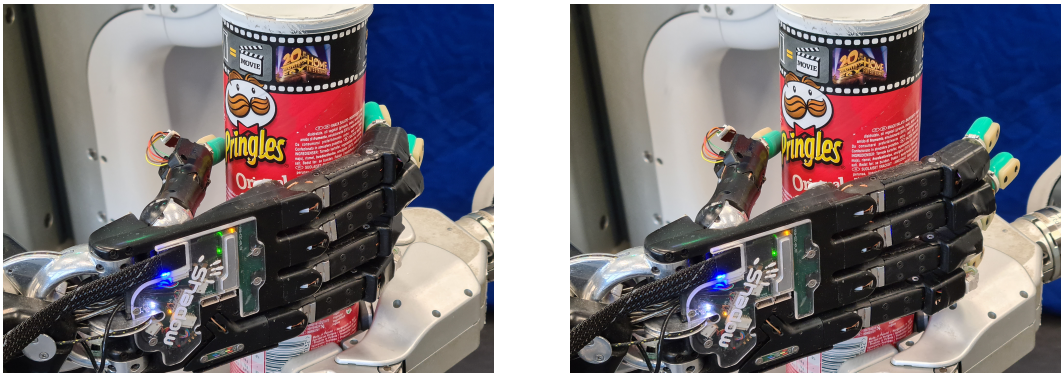


Figure 5.20.: Deteriorating behavior of the second training attempt. The model manages to make contact with the parallel fingers on the object after only 100 training steps, as shown on the left. On the right, the image shows how the model no longer made this contact after the training of 12000 steps.

in Figure 5.18. Such behavior is the opposite of a converging policy. To ensure that this was not a result of a training time that was too short, the training time for this initial test was extended to 24000 steps, twice the usual amount. As can be seen, the outcome did not change over time. For the entropy coefficient, Figure 5.19 shows a monotonic decline of the parameter of this attempt. A possible explanation for this phenomenon would be that the entropy coefficient induces too much exploration of worse actions, and the initial well-performing actions are becoming less likely to be executed again.

The training repeated with fixed poses for the grasp and handover poses to rule out that this structural variance in the pipeline caused the deteriorating behavior. Reward and average episode length showed the same deteriorating behavior as before. This deterioration is also visible when executing a grasp with the same model at an early and final timestep. As seen in Figure 5.20, the grasp of the early policy managed to get contact with the object. On the other hand, the final policy moved the parallel fingers by a negligible margin, making contact with only the thumb. The adaptable entropy coefficient also showed the same monotonic decline. These results indicate that the training parameters caused this behavior, not the pipeline structure. To further explore this behavior and to find a suitable entropy coefficient, the following training run started with a significantly lower entropy coefficient of 0.1. It continued to utilize the adaptable coefficient learning method from stable-baselines3. One further observation was the value of the average reward. Values between 3 and 5 for the reward are remarkably high, given that the final reward for a successful grasp could only have been 1.45. The difference must have been received as rewards through the closing reward for the intermediate steps, as stated in Equation 4.3. Since this might have incentivized the model to explore longer closing trajectories, the third training set this intermediate reward to 0, so only a successful or failed grasp provided a meaningful reward. However, the deteriorating behavior remained, even with this lower reward. Further, the adaptable entropy coefficient also had the same decline as previously. This decline indicates that the coefficient might still not be low enough, or the adaptable method of determining the coefficient might cause

issues.

The fourth training turned off the coefficient learning procedure and used a constant entropy coefficient of 0.05 instead. Correspondingly, the coefficient values for this and the following training attempts no longer show in Figure 5.19. Further, it scaled the reward at the end of a grasp attempt by the required timesteps to increase the incentive for fast closing motions. This scaling changes Equation 4.3 to

$$r_t = \begin{cases} r_b + r_con) * (1 - \frac{t}{t_{max}}), & \text{if } t = T_{final} \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

where t_{max} is the maximum limit of timesteps allowed for one grasping attempt. Finally, to determine if the state space causes training issues, the state space was reduced to just the one-hot encoding of the current object. Without changing information, the model should learn one best action for each object. As the results show, the policy stopped deteriorating over time. However, the average episode length and reward fluctuated greatly and depended on the recently grasped object. Without any meaningful state information, this is an expected behavior. Consequently, it only produced constant values as actions, seen in Figure 5.21. However, this setting showed that it is possible to learn a policy that does not deteriorate, even though it is primitive.

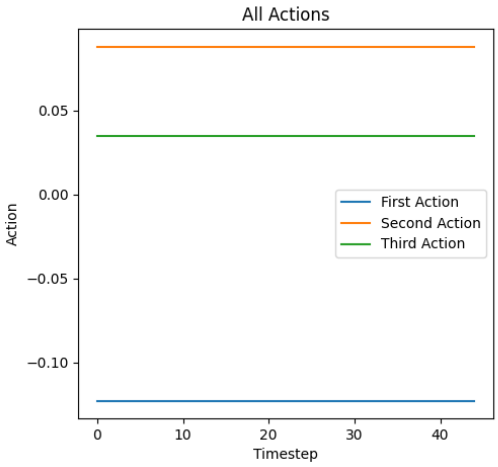


Figure 5.21.: Action values over all timesteps for an example grasp of the chips can. The used model only received the constant one-hot encoding as input.

Next, the fifth attempt keeps the time-scaled reward and restores the state space to the original setting. However, to test if training at all is possible with the whole variable pipeline, the entropy coefficient was set to 0, turning off the entropy exploration incentive. This coefficient drastically increases the training's susceptibility to the initial policy values. However, the initial reasonable performance of most previous models indicates that a suitable number of well-performing training examples from a randomly initialized policy is likely to be obtained.

As the results show, this was the case, and this training procedure showed the first stabilizing policy, where the average episode length and reward stayed constant without the previous deterioration. Therefore, to produce similar results with a model capable of some exploration to be less influenced by the initial policy values, a final training was conducted with a low, constant entropy coefficient of 0.001. As the training data depicts, the model performed similarly to the previous training test. Therefore, this ended the further optimization of the entropy coefficient. Overall, the training time for these attempts ranged between 1.4 and 3.5 hours for the standard training time and 5.0 hours for the double training time. These variations occurred because of two reasons. First, different numbers of hardware errors required different amounts of training restarts. Second, faster closing motions required more grasp tests, which consumed additional time. Nonetheless, the required time to train a model remained reasonable for all attempts, showing the viability of the training setup for real-world scenarios that require the constant presence of a human operator. The following section continues with the evaluation of this seemingly well-trained model.

5.2.2. Side Grasp

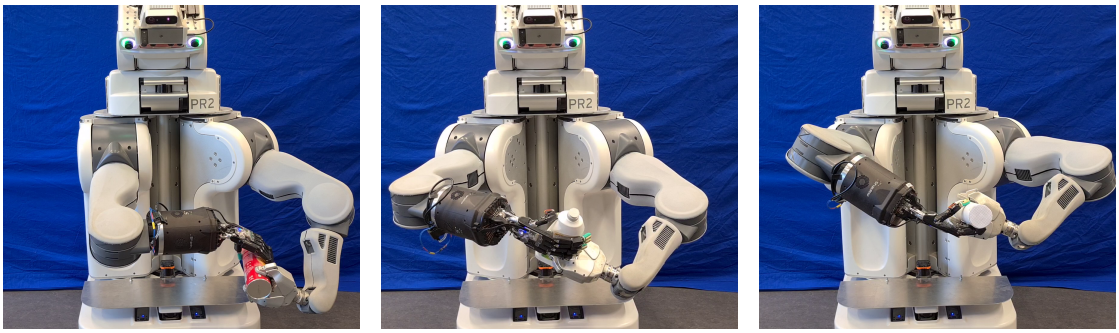


Figure 5.22.: Examples of side grasp handovers for each object.

The last model trained for hyperparameter optimization is also the final model for side grasps. As it showed suitable training performance, this section evaluates its performance as part of the entire handover pipeline. To this end, the pipeline had to hand over each object ten times, both with the model and the constant value hand closer as a baseline. Figure 5.22 shows examples of these handovers. An attempt was successful if the object stayed in the robot's hand for at least three seconds in a final pose after the handover. For examples of successful grasp attempts ending in this pose, refer to Figure 5.23. Further, Table 5.3 lists the results for all attempts. In these results, two things are notable at a glance. First, the model performed slightly worse than the baseline. Second, the bleach bottle performed notably worse than the other objects for both baseline and model.

There are some possible explanations for the first observation. First, both versions performed equally well for the paper roll. Next, the two failed grasping attempts for the chips can by the model happened because of an odd behavior of the policy for the can. Initially, all fingers were closing around the object, but after the ring and little finger had made con-

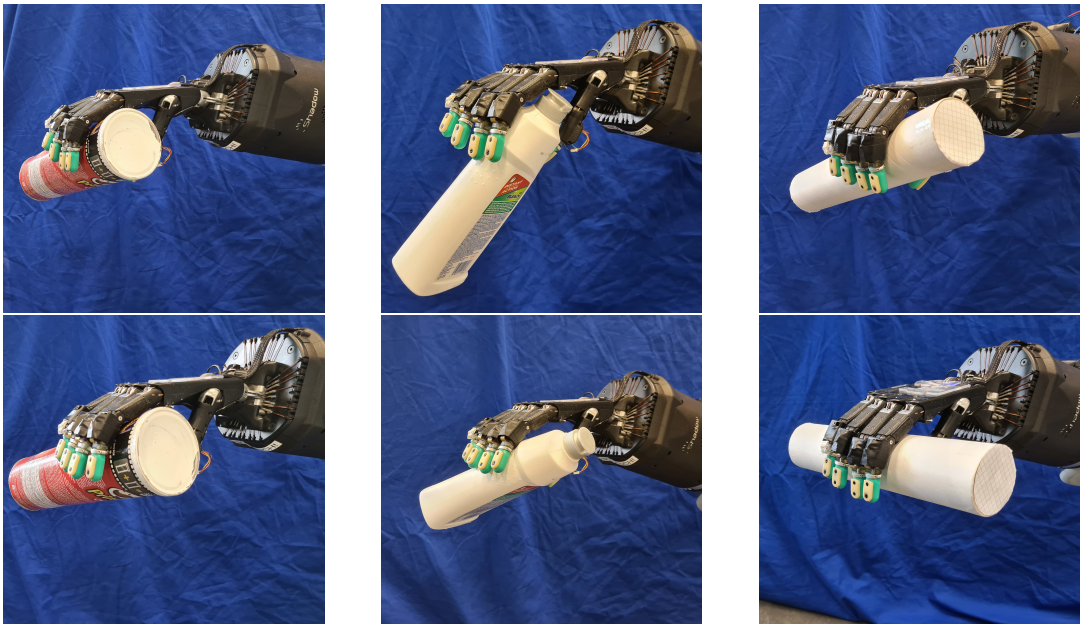


Figure 5.23.: Examples of successful side grasps for all three objects in the final evaluation pose. The top row shows grasps generated by the baseline, while the bottom row depicts model-generated grasps.

Table 5.3.: Success rates of pipeline executions for model and baseline generated side grasps. Each grasp counted as successful if the robot held the object for at least three seconds with its hand in the final pose.

	chips can	bleach bottle	paper roll
baseline	10/10	7/10	10/10
model	8/10	5/10	10/10

tact with the can, the speed of closure for the first and middle fingers slowed. Usually, they still managed to contact the can, but in some instances, they only did so after reaching the maximum step limit. In these instances, the object was more unstable and could fall out, as happened twice. One explanation would be that, due to still suboptimal hyperparameters, it explored this behavior during the end of the training time and needed more experience to counteract it completely.

Interestingly, the last three objects during training of this model contained the chips can twice. As Figure 5.24 shows, the model had no such behavior before these last 3000 training steps. Together with the slight increase of the average episode length for this model visible in Figure 5.17 indicates that it might deteriorate with further training again, requiring further hyperparameter optimization in scenarios with longer training time. However, this also indicates the model's ability to learn different behaviors for different objects, as no such issues occurred when handling the other two objects. It is remarkable due to the similarity in shape

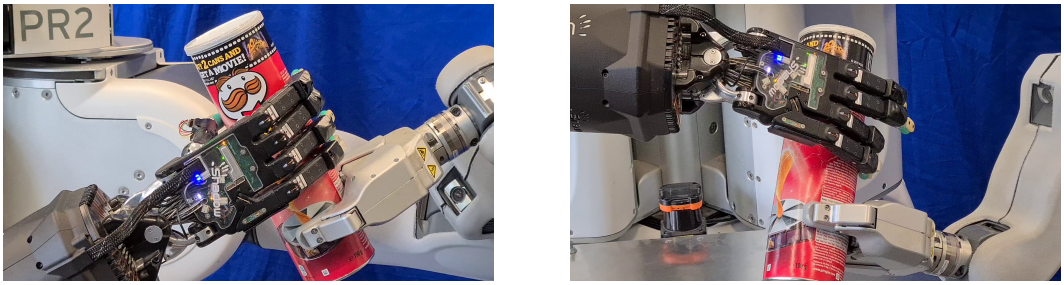


Figure 5.24.: Final model grasping behavior for the chips can. The first and middle fingers slowed during the closing motion, sometimes failing to contact the can. The right image shows an example of this. However, the left image shows the model after only 9000 training steps, not demonstrating this behavior.

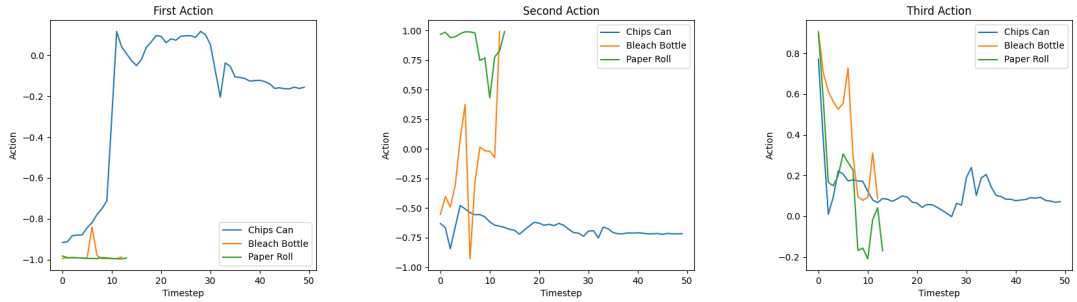


Figure 5.25.: The development of model-generated actions during an example side grasp for all three objects. From left to right, show the plots of this development for the network's first, second, and third output dimensions.

between the chips can and the paper roll, as they both are cylindrical with different radii.

This difference is also visible when looking at the development of action outputs for the different objects in Figure 5.25. The model made contact with all fingers for the bleach bottle and paper roll before the maximum step limit, so the output stopped early. Due to the previously mentioned chip can issue, the model did not stop early for it. Since the pipeline normalizes the first and second action between 0 and -1 before generating the joint configuration through the synergies, 1 in the graphs corresponds to no weight for that synergy. At the same time, -1 is the maximum weight for a closing motion. The first synergy corresponds to a general closing motion for all four parallel fingers without moving the thumb. As the plots show, the model outputs -1 for the bleach bottle and the paper roll almost during the grasping motion. For the chips can, the output changes to 0 around the time it finishes for the other two objects, meaning it only continues to close the fingers at half the speed as previously. Since it also started lower for the chips can, this might explain why it did not manage to make contact with the object in time.

The model also had an almost constant output for the chips can as the second action for a strong closing motion. This synergy moves the thumb. Interestingly, this output remained

high even though the thumb made contact with the object. One theory is that the thumb has already started too close to the chips can, so the change in effort was not high enough to pass the threshold. This observation might be an explanation for the chips can behavior as well. For the other two objects, this value switched to 1 at the end of the closing motion, indicating that the thumb no longer had to be moved. It was already close to 1 at the start with the paper roll, only slightly changing during the grasping motion. For the bleach bottle, it started at a lower value and changed to 1 significantly later. One explanation is that the thumb already starts closer to the paper roll than to the bleach bottle, so it has to move less overall. The third action has no normalization, as it has no distinct closing direction, so 0 means almost no change, while 1 and -1 are the maximum changes in the two directions, respectively. Here, the output for all three objects is similar. It changes relatively fast to a low value of around 0, and for the chips, it can stay at that value while the grasping motion terminates earlier for the other two objects. Seemingly, it is not significant for the grasping motion of the side grasp, as expected.

For the bleach bottle, the difference in performance is not as different as the result seems. In both cases, the bleach bottle is often twisted in the hand, either directly after the gripper retracted from the object or during the three-second holding in the final pose. For the baseline, this twisting sometimes pushed the top of the bottle against the side of the hand to prevent further twisting and falling out, as in the example grasp for the baseline with the chips can in Figure 5.23. About three of the successes for the baseline encountered this scenario. Without these, both options would have performed similarly on the bleach bottle.

This observation leads to the second observation on why the bleach bottle performed significantly worse than the other two objects. First of all is the material. The bottle is significantly smoother than the other two objects. This smoothness can lead to the fingers slipping, especially if the contact between fingers and object is not at the fingertips, where the rubbery BioTac sensors can apply increased friction, but at either of the other two finger links.

Figure 5.26.: The left image shows the orientation of the bleach bottle with which the model trained. In the right image, the bleach bottle's orientation is rotated by 180° to test the performance with this alternative insertion method.

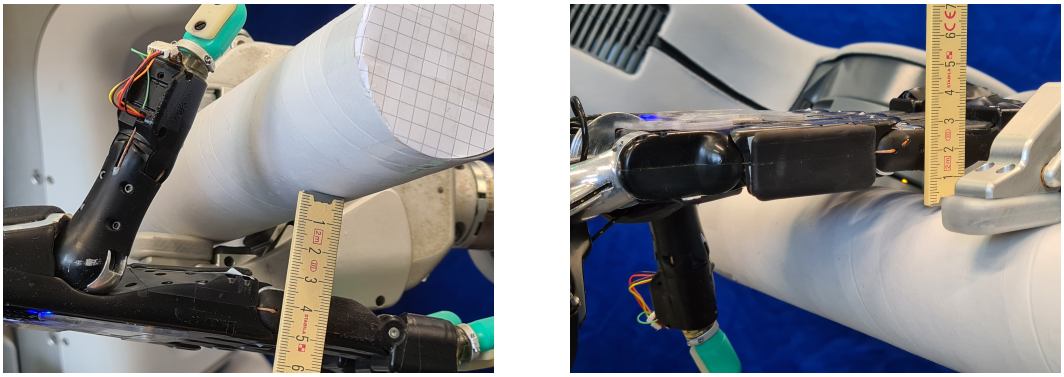


Figure 5.27.: The variable distances of handover poses where the hand is above or below the object.

Next is also the more complex shape. Unlike the other two objects, which are cylindrical, the bleach bottle is less uniform and symmetric. This shape makes the relative grasp pose and its influence on the outcome of the grasp more significant. Another test rotated the bleach bottle by 180°, as seen in Figure 5.26. However, tested with the model, this orientation only managed to perform three successful grasps out of ten. Notable here is that the model managed to produce grasps where all fingers made contact with the object, even though it never trained with this new orientation.

There is one other issue regarding the grasp pose. Besides the variance introduced by manually inserting the object in the left gripper and generating the grasp pose based on the point cloud reading, another factor significantly influences the relative grasp pose. This factor is whether the selected handover pose has the hand above or below the object. The most common handover pose had the object completely horizontal. When the object faces towards the robot, the hand hovers over it. While the object points away from the object, the hand is below it. All successful grasps of the bleach bottle had the hand above, while most failures had the hand below the object. As seen in Figure 5.27, the hand is significantly closer to the object when it is above the object and farther away when below, even though the relative pose between gripper and hand is the same according to both targeted pose and reported robot state. One explanation for this would be a gravity-related calibration error in the right arm. Since the Shadow hand is not the original lower arm on the PR2, some calibration errors likely remained after the adaptation to the new hardware. While it might be negligible in most scenarios, in this situation, where the hand-to-object pose rotates by 180°, this difference can be noticeable in the distance between hand and object. For the other two objects with more uniform shapes, this influence also does not influence the result too much. However, for the bleach bottle, where more precision is required, this can significantly influence the success rate.

Alternatively, the depth information generated using the point cloud might be less precise than expected. As the side of the object is not visible from the static observation pose, all object depth information has to come from viewing the top of the object. Consequently, the pipeline might choose whether the hand is above or below the object by this variation. This

theory would flip the causality of the observed correspondence between distance and hand being below or above the object. Under that assumption, having a point cloud reading that causes the hand to be closer to the object causes the pipeline to choose handover poses where the hand is above the object and vice versa. Theoretically, such issues are why performing the learning process on a real-world platform can be beneficial compared to simulation training. In this case, however, the model did not counteract this difference, and it is difficult to tell if a simulator-trained model would handle this situation similarly or worse.

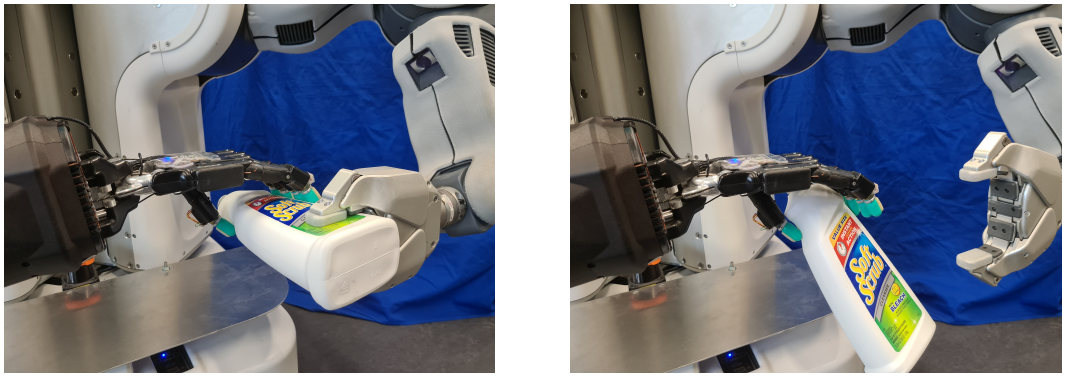


Figure 5.28.: An example of the object twisting in the hand after the gripper retracts.

One final observation is the twisting motion already mentioned. While most prominent for the bleach bottle, it also happened less severely for the other two objects. The reason for this twisting motion is the ability of the object to rotate more freely when the gripper retracts from the object. Grasp quality testing in the pipeline is only performed through a check for translational slippage, not rotational movements. For example, when the parallel fingers and the thumb enact forces on an object that are not directly facing each other, they create a rotational movement of the object. With this motion, the object can rotate into a pose where contact with the fingers is lost, and it can fall out of the grasp. While this is less likely to happen for the chips can and the paper roll due to their symmetry, this is much more likely to happen with an asymmetric object such as the bleach bottle. For the model training, this is especially problematic since grasps that test as stable during training can now be unstable during execution. Therefore, the model cannot learn a policy to counteract this issue. This inability is another reason why the model underperformed with the bleach bottle.

Further discussion of solutions to these problems continues in section 6.2.

5.2.3. Top Grasp

To allow for grasping the models with the top grasp requires training a new model. This training uses the same hyperparameters as the model for side grasps. The exception is the reward function. As the top grasp is more likely not to allow all fingers to contact the object, the reward is no longer time-scaled. For this reason, the average episode length will remain at or close to the maximum step limit. Conversely, the reward fluctuates significantly, as seen in Figure 5.29. While not optimal, that behavior likely results from the more complex nature

Figure 5.29.: Average reward for the top grasp model during training.

of the top grasp. Interestingly, the significant reduction between steps 8000 and 9000 looks like deteriorating behavior, but the reward climbs steadily afterward again. Consequently, this section continues to use this model for further evaluation even with that negative spike.

The model required training for 2.25 hours. After the training, the evaluation uses the same method described for the side grasp with examples of the performed handovers visible in Figure 5.30. Similarly, Figure 5.31 shows examples of successful handover attempts, while Table 5.4 lists the success rates of all grasping attempts. The significant improvement of the model and baseline for the bleach bottle is notable. Also, the model performed similarly to the baseline. That baseline shows a perfect success rate.

One explanation for the remaining difference between the model and baseline might be the

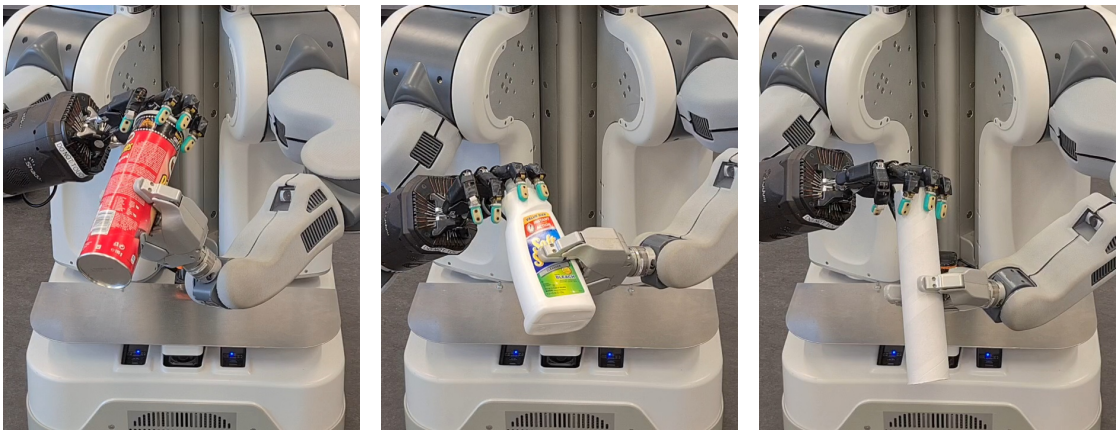


Figure 5.30.: Examples of top grasp handovers for each object.

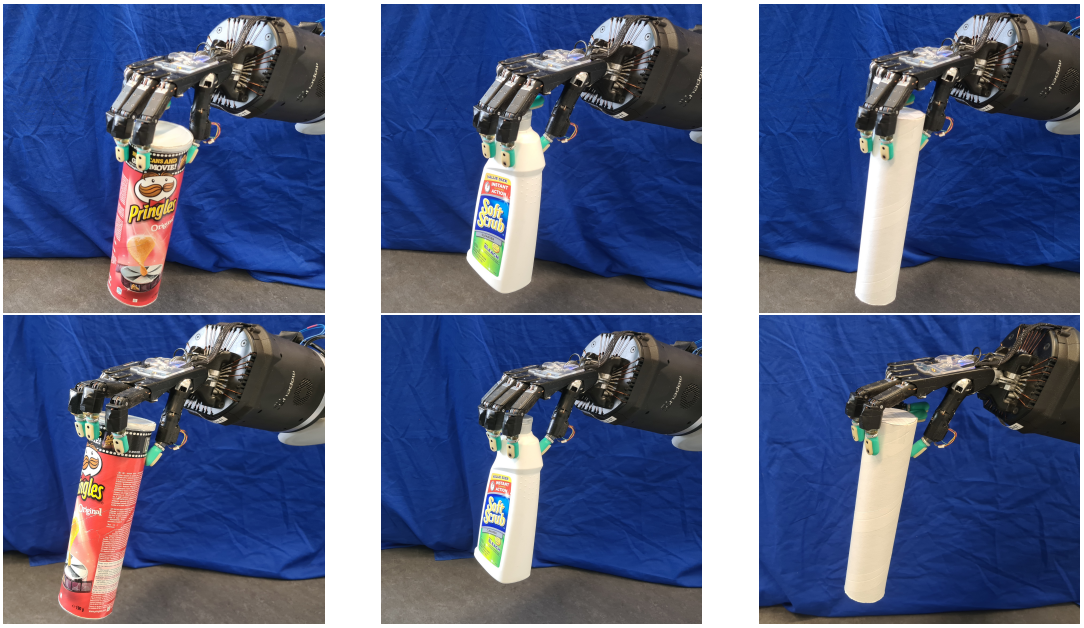


Figure 5.31.: Examples of successful top grasps for all three objects in the final evaluation pose. The top row shows grasps generated by the baseline, while the bottom row depicts model-generated grasps.

Table 5.4.: Success rates of pipeline executions for model and baseline generated top grasps. Each grasp counted as successful if the robot held the object for at least three seconds with its hand in the final pose.

	chips can	bleach bottle	paper roll
baseline	10/10	10/10	10/10
model	9/10	9/10	9/10

significantly more angled fingers for model-generated grasps. As visible in Figure 5.31, the model learned to close the second joints of the fingers more before making contact with an object. This behavior can result in the fingers missing the object during closing if the hand is higher above the object than usual. The distance between the object and the hand varied greatly during training and testing. While it should have remained constant, it indicates issues in the consistency of height estimation from the point cloud. Such differences might occur when the human operator inserts the object slightly tilted in the gripper, resulting in a higher point from one of the tilted object corners. Unlike during the side grasps, no pattern emerged for these height differences during testing, increasing the likelihood of this hypothesis about inconsistent readings extracted from the point cloud. While calibration errors might still contribute to the findings for the side grasp, they are less likely to be the main reason for the inconsistency issues.

However, this also highlights an achievement of the model. The model adapted to this

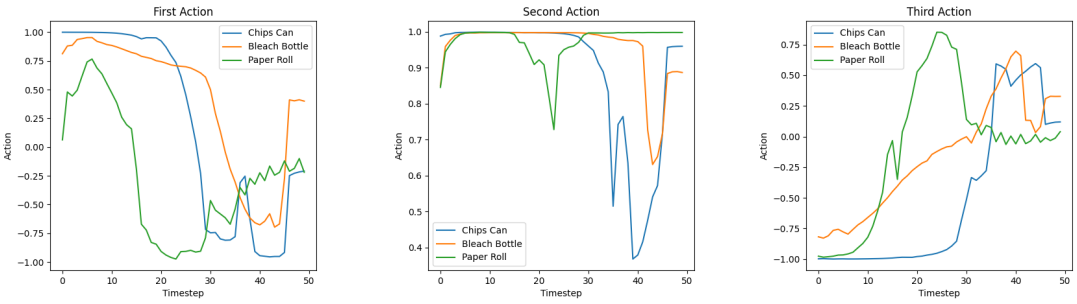


Figure 5.32.: The development of model-generated actions during an example top grasp for all three objects. From left to right, show the plots of this development for the network's first, second, and third output dimensions.

circumstance by curling the fingers before closing. This motion also shows in the actions generated by the model, seen in Figure 5.32. They follow the same rules as those for the side grasp. Correspondingly, the first action shows that the model initially generated values close to 1 for all three objects, so the closing motion for the fingers is minimal at the start. Only after several timesteps does the model change this value significantly, resulting in a strong closing motion for the parallel fingers.

The same pattern emerged for the second action, so the thumb closed together with the parallel fingers after similar timesteps. On the other hand, the third action started with a high negative value, corresponding to the curling motion of the fingers without closing them. The third action also changed when the other actions started their closing motion. This change initially moved the values to higher positive values, likely as further adaptation during the closing motion, after which they fell toward zero.

Consequently, the results indicate the ability of this training process to adapt more complex solutions. Further, it shows that the hand synergies allow for a significantly different grasping behavior than closing the fingers with a constant motion, even with the applied restrictions.

The difference in success rate for the bleach bottle likely results from the difference in orientation for the finger contact points. While the side grasp at first looks like a power grasp, the object does not have contact with the hand palm. Consequently, it is a precision grasp, similar to the top grasp. Therefore, it cannot counteract the forces applied by the thumb and parallel fingers with the palm. On the smaller sides of the bleach bottle, the requirement to have the forces from the thumb and parallel fingers cancel each other out without applying a rotational movement is significantly more complex than on the larger sides used by the top grasp.

Overall, the results show the possibility of applying this training method to other grasp types and the possibility for such a model to learn more complex behaviors. Further discussion of these results follows in section 6.2.

6. Discussion

This chapter discusses the results of this thesis and possible solutions to the issues that occurred. The first part involves the general structure of the pipeline as well as the corresponding workspace analysis. Afterward, the second part discusses findings from the real-world RL attempt to learn grasping strategies.

6.1. Pipeline Architecture

As mentioned above, the handover pipeline works successfully but has limitations. First of all, the grasp poses selection. This process involves getting a point cloud of the object and determining where to grasp the object. Without object exploration, the point cloud only provides information from one viewpoint, disregarding object parts outside the predetermined filter area. While being a limitation, the latter part is a reasonable assumption for simple objects. However, for objects that, for example, do not fit entirely into the gripper, this can produce issues even with the simple grasp selection process. Even more limiting is the point cloud reading from a fixed angle. Since the current grasp pose selection relies on the depth information of the object, any object with a more complex depth profile will likely result in the hand being too far away from the object or colliding with it during the handover process. As the results have shown, this issue has already occurred with the selected object set.

Additionally, the grasp pose selection only uses the point cloud's minimum and maximum values, with fixed offsets regardless of the object shape and without any outlier detection. A human also previously set these offsets, so this significantly limits generalizability. While this simple approach works for this thesis, due to the limited objects and as a proof of concept for the general pipeline architecture, this is one of the most significant limitations of this work in the future. Possible solutions could involve a more sophisticated object exploration to get more information about the object's shape. Further, by utilizing a neural network for grasp estimation, i.e., PointNetGPD [52] or similar architectures, a more generalizable grasp pose generation might be possible. However, these architectures would need to be adapted to account for the shape of the Shadow hand during grasp pose generation. Also, the desired grasp type has to be set by a human at the start of the handover process. However, this issue might be easier to solve by deciding the grasp type based on the point cloud shape. For example, switching between this thesis's side and top grasp could depend on whether the size of the point cloud above the gripper is large enough to support a side grasp.

Another point is the issue of handover pose selection. While this works very well in the pipeline, due to the offline computed workspace analysis, it only works when the grasp types are predetermined. If the pipeline should work with more grasp types, a previous workspace analysis would need to be computed previously for every new grasp type to enable efficient

handover pose sampling. Otherwise, if the grasp were not analyzed beforehand, only sampling over the entire possible workspace would be a suitable option. One way to solve this issue would be to take a closely related grasp as a basis for the optimal sampling region, but this would require such a grasp to have such an analysis already.

Finally, there is the issue of expandability. The results show that the pipeline can perform different grasp types on varying objects. While extending the pipeline seems possible, this discussion is part of the following section. Another part of expandability is the possibility of applying the pipeline to different robot systems. Such an expansion would be rather challenging. The system would need very similar capabilities. For example, it would require the same manipulator setup of having a two-finger gripper and a humanoid hand. Such a setup is uncommon. Further, similar sensor setups are necessary to perceive the point cloud, finger efforts, and forces applied to the two-finger gripper arm. While making expansion theoretically possible, finding alternatives for these requirements would greatly benefit the use of this thesis's findings.

6.2. Training

The initial challenge for the model training was the selection of appropriate hyperparameters. While it shows that training using SAC allowed most hyperparameters to remain at their default value, it took several training attempts and human ingenuity to choose an appropriate value for the final hyperparameter, the entropy coefficient. Even with this process, some remaining deteriorating behavior indicates that it did not find an optimal parameter. Without a feasible automatic optimization for selecting this hyperparameter, future pipeline, and real-world system RL adaptations remain challenging. Possible solutions for this issue will be required, especially when the complexity and training time increase for future tasks.

Next was the evaluation of the trained model. As previously explained, the results show the possibility of training a working model on a real-world system. Compared to the previous work by Liang et al. [22], this thesis achieved similar results. They reported a success rate of 50% for the bleach bottle and 100% for the chips can. This result highlights that the issues with the bleach bottle are not exclusive to the thesis. While this thesis achieved a success rate of 90% for the bleach bottle with the top grasp, this thesis has fixed grasp types, while they were flexible in the work by Liang et al. While Haschke et al. [39] did not evaluate their pipeline on the bleach bottle, they evaluated it on various cylindrical objects, including the chips can. They report a success rate of 73.3% to 100%, with the latter one being the success rate for the chips can. While the model in this thesis performed worse for the chips can, the success rate for the chips can and the paper roll, both cylindrical objects, also lie in that range. Consequently, the trained model performs competitively with other work in the field.

However, as the results show, this model's performance remained below the baseline, although it managed to perform similarly. Multiple explanations already came up in subsection 5.2.2. Another issue raised was the performance difference for varying handover poses. While the explanations likely involve variations during grasp pose selection and possible calibration errors, this should have been a case where training on the real-world system can learn

to counteract this. However, the model only partially learned such a compensation.

One reason for this is likely the limitations applied to joint control for the model. For example, if the model could have also moved the wrist joints, it might have been able to adapt the grasp pose slightly through wrist motions to counteract the distance difference. In practice, this would cause other issues, e.g., preventing the hand from colliding with the object during these wrist adjustments. This issue is one of the reasons why these limitations exist in this thesis. Another reason for this and other joint limitations is the potential damage to the hardware during training. The restrictions on the fourth joints of the parallel fingers or the wrist joints are difficult to lift. To prevent the hand from colliding with the gripper during the handover or the fingers from colliding with itself without these restrictions, every step must be performed with collision-aware planning. This planning would significantly increase the execution time for any grasp attempt, causing issues with the practicality of real-world RL. Further, this would require reliable and precise calibration of the entire system, which, as shown, can be an issue. While the compliance of the robot system should prevent significant immediate damage, for example, fingers colliding with the sensors mounted on top of other fingers is very likely to cause long-term issues.

Next is the observed twisting motion of the objects after the gripper retracts. As mentioned, with only a translational slippage check for estimating the success or failure of a grasp, the model cannot learn methods to prevent this twisting. However, extending the grasp stability estimation to include rotational movements is challenging. First, there is no similar sensor whose output can determine rotational twisting by one movement. Instead, a combination of force readings from different directions would need consideration with a sensor like the force sensor. Sensor readings from additional sensors, such as pressure sensors in the gripper fingertips, would likely also need consideration to get an estimation. Alternatively, additional grasp quality metrics could be considered, such as geometric calculations based on the contact positions on the object point cloud. However, depending on the complexity of such an estimation, it would require an individual research project.

This complexity is especially the case when considering that the quality estimation still needs to work in the same self-supervised capacity as the simplified metric of this thesis to allow for real-world RL. For the same reason, testing the grasp by releasing the gripper is not feasible, as it would require constant human labeling during training. Another improvement that can reduce this object twisting is further encouraging contact of the fingertips with the object. With the rubber coating on the BioTac sensors, these grasps are likely to reduce object slippage, especially for objects with smooth surfaces. One way might be to change the reward to provide the additional contact reward only when the fingertips have made contact with the object instead of any part of the finger.

Another significant issue in this thesis is the limited scope of objects and corresponding grasp poses. Only three objects with similar shapes were used and evaluated for only two predefined grasp types. These are also the only objects used for testing the model and general pipeline. This limited scope was to first test the idea of RL on a real-world system in a simplified setup and figure out potential sources of failure. As has been discussed, these emerging challenges were plentiful, and further exploration of them is necessary before expansion to more complex settings. That still leaves the question of how likely and feasible such an expansion would be.

First, as discussed about the pipeline structure, expanding to more object and grasp types would require a more sophisticated grapes selection strategy.

Additionally, generalizing to various objects would require replacing the current one-hot encoding of the object in the state space. This replacement might be possible by an offline trained encoder for the point cloud input. However, such an expansion also makes it likely to increase the size of the state space and, therefore, the training time. With more varying objects, this might already be the case, so any increase in the complexity of the model needs to be monitored according to the increase in training time to keep it feasible for a real-world training application.

7. Conclusion

The two focus points of this thesis were the following. First, how to implement a bimanual handover pipeline on the PR2 platform used in this thesis that utilizes multi-modal feedback but no previous data about the object's shape or structure. As shown, this challenge was solved successfully with the proposed pipeline. It manages to perform bimanual handovers for multiple objects. No explicit shape information of the object, such as a previously recorded point cloud from multiple views or an entire mesh of the object or any other objects, is required. The pipeline generates information about the object through an online perceived point cloud. Further, the pipeline uses this information with other modalities, specifically joint efforts and force feedback, to perform a handover. However, there are limitations to this pipeline and possible improvements. These involve needing a more sophisticated grasp of pose selection and object exploration strategy.

One additional aspect of this topic was the analysis of the workspace for the handover task. This thesis provided an analysis of the grasp types used. The analysis provided an optimal region to sample handover poses during online pipeline execution. Further, it showed that these regions remain relevant even for minor variations like the ones introduced by varying object point clouds. However, it also highlighted the requirement to repeat such an analysis for different grasp types, which makes expanding the pipeline to other grasp types challenging.

The second topic was whether it is feasible to use an RL-based approach to learn how to perform the grasping part of the handover, which is trained entirely on the real-world system. Results also show the feasibility of this idea. While the trained models showed slightly worse performance than the tested baseline, they learned how to solve the grasping task successfully. Further, they showed the possibility of generating more complex behavior than the baseline and adapting to solve challenges arising from the real-world application. However, the results also highlighted existing issues and limitations. Two significant ones are the issue of hyperparameter selection and the need for further testing on a more extensive variety of objects and grasp types.

Future work could explore multiple avenues to address the issues explored. One way is to incorporate further modalities with the pipeline, such as the BioTac sensors at the hand's fingertips or pressure sensors on the two-finger gripper. With this additional input, strategies to generate better grasps or better grasp quality metrics might be possible. Another direction is to use object exploration strategies for more robust information about the object during handover. Alternatively, future work could extend the pipeline's capabilities further, such as enabling the initial grasp of the object itself or attempting functional grasps with the hand. Other directions involve pre-training the model, either in simulation or through human demonstration, and then using the existing pipeline for further refinement.

Bibliography

[1] Valerio Ortenzi et al. "Object Handovers: A Review for Robotics". In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1855–1873. DOI: 10.1109/TRO.2021.3075365.

[2] Christian Smith et al. "Dual arm manipulation—A survey". In: *Robotics and Autonomous Systems* 60.10 (2012), pp. 1340–1353. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2012.07.005.

[3] Adrià Colomé and Carme Torras. "Reinforcement learning of bimanual robot skills". In: Springer, 2020, p. 15.

[4] Chunmiao Yu and Peng Wang. "Dexterous Manipulation for Multi-Fingered Robotic Hands With Reinforcement Learning: A Review". In: *Frontiers in Neurorobotics* 16 (2022). ISSN: 1662-5218. DOI: 10.3389/fnbot.2022.861825.

[5] Julian Ibarz et al. "How to train your robot with deep reinforcement learning: lessons we have learned". In: *The International Journal of Robotics Research* 40.4-5 (2021), pp. 698–721. DOI: 10.1177/0278364920987859.

[6] Willow Garage. *PR2 User Manual*. Accessed on February 27th, 2024. 2012. URL: https://www.clearpathrobotics.com/assets/downloads/pr2/pr2_manual_r321.pdf.

[7] Shadow Robot Company. *Shadow Dexterous Hand Technical Specification*. Accessed on March 29th, 2024. 2021. URL: https://www.shadowrobot.com/wp-content/uploads/2022/03/shadow_dexterous_hand_e_technical_specification.pdf.

[8] Kenneth J. Waldron and James Schmiedeler. "Kinematics". In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Cham: Springer International Publishing, 2016, pp. 11–36. ISBN: 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1_2.

[9] Jacques Denavit and Richard S Hartenberg. "A kinematic notation for lower-pair mechanisms based on matrices". In: (1955).

[10] Serdar Kucuk and Zafer Bingul. "Robot Kinematics: Forward and Inverse Kinematics". In: *Industrial Robotics*. Ed. by Sam Cubero. Rijeka: IntechOpen, 2006. Chap. 4. DOI: 10.5772/5015.

[11] K.C. Gupta. "On the Nature of Robot Workspace". In: *The International Journal of Robotics Research* 5.2 (1986), pp. 112–121. DOI: 10.1177/027836498600500212.

[12] A Kumar and KJ Waldron. "The workspaces of a mechanical manipulator". In: (1981).

[13] Tsuneo Yoshikawa. "Manipulability of Robotic Mechanisms". In: *The International Journal of Robotics Research* 4.2 (1985), pp. 3–9. DOI: 10.1177/027836498500400201.

[14] Nikolaus Vahrenkamp et al. "Manipulability Analysis". In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. 2012, pp. 568–573. DOI: 10.1109/HUMANOIDS.2012.6651576.

[15] Franziska Zacharias, Christoph Borst, and Gerd Hirzinger. "Capturing robot workspace structure: representing robot capabilities". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, pp. 3229–3236. DOI: 10.1109/IROS.2007.4399105.

[16] Máximo A Roa and Raúl Suárez. "Grasp quality measures: review and performance". In: *Autonomous robots* 38 (2015), pp. 65–88.

[17] Brayan S. Zapata-Impata, Pablo Gil, and Fernando Torres. "Tactile-Driven Grasp Stability and Slip Prediction". In: *Robotics* 8.4 (2019). ISSN: 2218-6581.

[18] Rocco A. Romeo and Loredana Zollo. "Methods and Sensors for Slip Detection in Robotics: A Survey". In: *IEEE Access* 8 (2020), pp. 73027–73050. DOI: 10.1109/ACCESS.2020.2987849.

[19] Takashi Maeno, Shinichi Hiromitsu, and Takashi Kawai. "Control of Grasping Force by Estimating Stick/Slip Distribution at the Contact Interface of an Elastic Finger Having Curved Surface". In: *Journal of the Robotics Society of Japan* 19 (Jan. 2001). DOI: 10.7210/jrsj.19.91.

[20] Marco Santello, Martha Flanders, and John F Soechting. "Postural hand synergies for tool use". In: *Journal of neuroscience* 18.23 (1998), pp. 10105–10115.

[21] Matei T. Ciocarlie and Peter K. Allen. "Hand Posture Subspaces for Dexterous Robotic Grasping". In: *The International Journal of Robotics Research* 28.7 (2009), pp. 851–867. DOI: 10.1177/0278364909105606.

[22] Hongzhuo Liang et al. "Multifingered Grasping Based on Multimodal Reinforcement Learning". In: *IEEE Robotics and Automation Letters (RA-L)* 7.2 (2022), pp. 1174–1181. DOI: 10.1109/LRA.2021.3138545.

[23] Alexandre Bernardino et al. "Precision grasp synergies for dexterous robotic hands". In: *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2013, pp. 62–67. DOI: 10.1109/ROBIO.2013.6739436.

[24] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd ed. MIT Press, 2018.

[25] David Valencia et al. "Comparison of Model-Based and Model-Free Reinforcement Learning for Real-World Dexterous Robotic Manipulation Tasks". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 871–878. DOI: 10.1109/ICRA48891.2023.10160983.

[26] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 1861–1870.

[27] Samarth Brahmbhatt et al. "ContactGrasp: Functional Multi-finger Grasp Synthesis from Contact". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 2386–2393. DOI: 10.1109/IROS40897.2019.8967960.

[28] Miao Li et al. "Dexterous grasping under shape uncertainty". In: *Robotics and Autonomous Systems* 75 (2016), pp. 352–364. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2015.09.008.

[29] Maximo A. Roa et al. "Power grasp planning for anthropomorphic robot hands". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 563–569. DOI: 10.1109/ICRA.2012.6225068.

[30] Lin Shao et al. "UniGrasp: Learning a Unified Model to Grasp With Multifingered Robotic Hands". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2286–2293. DOI: 10.1109/LRA.2020.2969946.

[31] Umit Rusen Aktas et al. "Deep Dexterous Grasping of Novel Objects From a Single View". In: *International Journal of Humanoid Robotics* (2022). DOI: 10.1142/s0219843622500116.

[32] Gabriel Dulac-Arnold et al. "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis". In: *Machine Learning* 110.9 (2021), pp. 2419–2468.

[33] A. Rupam Mahmood et al. "Benchmarking Reinforcement Learning Algorithms on Real-World Robots". In: *Proceedings of The 2nd Conference on Robot Learning*. Ed. by Aude Billard et al. Vol. 87. Proceedings of Machine Learning Research. PMLR, 2018, pp. 561–591.

[34] Tuomas Haarnoja et al. *Soft Actor-Critic Algorithms and Applications*. 2019. arXiv: 1812.05905 [cs.LG].

[35] Shixiang Gu et al. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 3389–3396. DOI: 10.1109/ICRA.2017.7989385.

[36] Yunfei Li et al. "Efficient Bimanual Handover and Rearrangement via Symmetry-Aware Actor-Critic Learning". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 3867–3874. DOI: 10.1109/ICRA48891.2023.10160739.

[37] Jean-Philippe Saut et al. "Planning pick-and-place tasks with two-hand regrasping". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 4528–4533. DOI: 10.1109/IROS.2010.5649021.

[38] Silvia Cruciani et al. *Dual-Arm In-Hand Manipulation and Regrasping Using Dexterous Manipulation Graphs*. 2019. arXiv: 1904.11382 [cs.RO].

[39] Robert Haschke, Guillaume Walck, and Helge Ritter. "Geometry-Based Grasping Pipeline for Bi-Modal Pick and Place". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 4002–4008. DOI: 10.1109/IROS51168.2021.9635981.

[40] U. Pattacini G. Vezzani M. Regoli and L. Natale. "A novel pipeline for bi-manual handover task". In: *Advanced Robotics* 31.23-24 (2017), pp. 1267–1280. DOI: 10.1080/01691864.2017.1380535.

[41] Benjamin Balaguer and Stefano Carpin. "Bimanual regrasping from unimanual machine learning". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 3264–3270. DOI: 10.1109/ICRA.2012.6225095.

[42] Dmytro Pavlichenko et al. "Autonomous Bimanual Functional Regrasping of Novel Object Class Instances". In: *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*. 2019, pp. 351–358. DOI: 10.1109/Humanoids43949.2019.9035030.

[43] Nikolaus Vahrenkamp et al. "Humanoid Motion Planning for dual-arm manipulation and re-grasping tasks". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 2464–2470. DOI: 10.1109/IROS.2009.5354625.

[44] Morgan Quigley. "ROS: an open-source Robot Operating System". In: *IEEE International Conference on Robotics and Automation*. 2009.

[45] David Coleman et al. *Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study*. 2014. arXiv: 1404.3785 [cs.RO].

[46] Philipp Ruppel et al. "Cost Functions to Specify Full-Body Motion and Multi-Goal Manipulation Tasks". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 3152–3159. DOI: 10.1109/ICRA.2018.8460799.

[47] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. "The Open Motion Planning Library". In: *IEEE Robotics & Automation Magazine* 19.4 (Dec. 2012). https://ompl.kavrakilab.org, pp. 72–82. DOI: 10.1109/MRA.2012.2205651.

[48] J.J. Kuffner and S.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. 2000, 995–1001 vol.2. DOI: 10.1109/ROBOT.2000.844730.

[49] Martin Pecka. *Robot Body Filter*. Accessed on March 10th, 2024. URL: https://github.com/peci1/robot_body_filter.

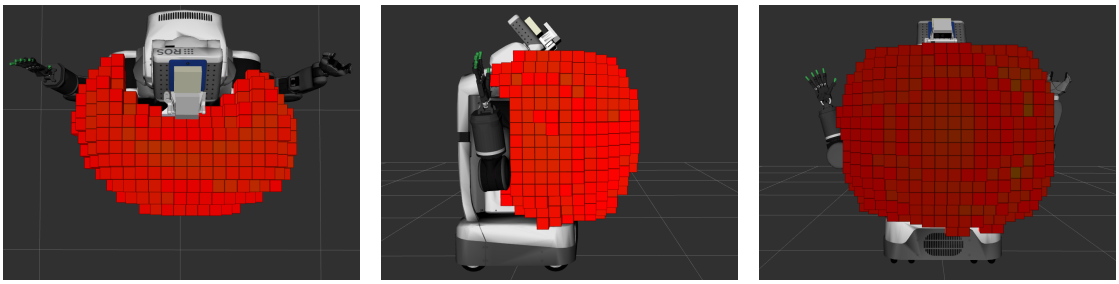
[50] Antonin Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8.

[51] Berk Calli et al. "Benchmarking in Manipulation Research: Using the Yale-CMU-Berkeley Object and Model Set". In: *IEEE Robotics & Automation Magazine* 22.3 (2015), pp. 36–52. DOI: 10.1109/MRA.2015.2448951.

[52] Hongzhuo Liang et al. "PointNetGPD: Detecting Grasp Configurations from Point Sets". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2019. DOI: 10.1109/icra.2019.8794435.

A. Workspace Analysis

A.1. Side Grasp



(a) Top (b) Side (c) Front

Figure A.1.: Additional views of all valid solutions for the x-shifted side grasp.

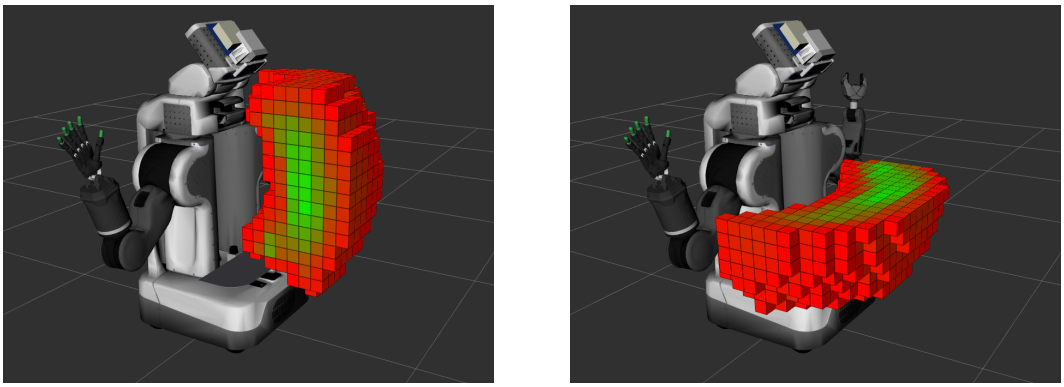
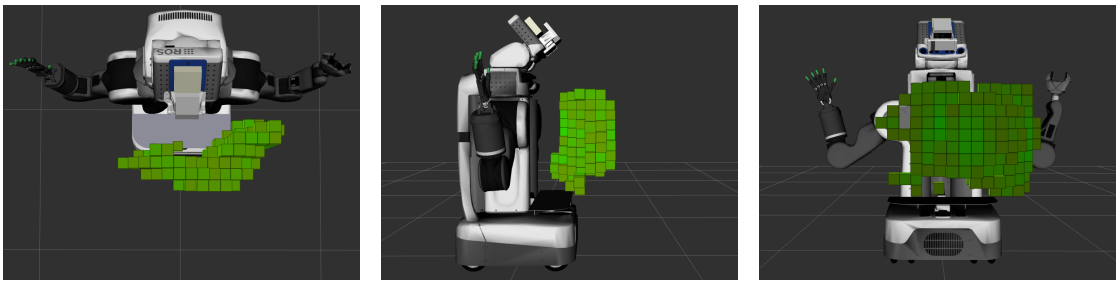
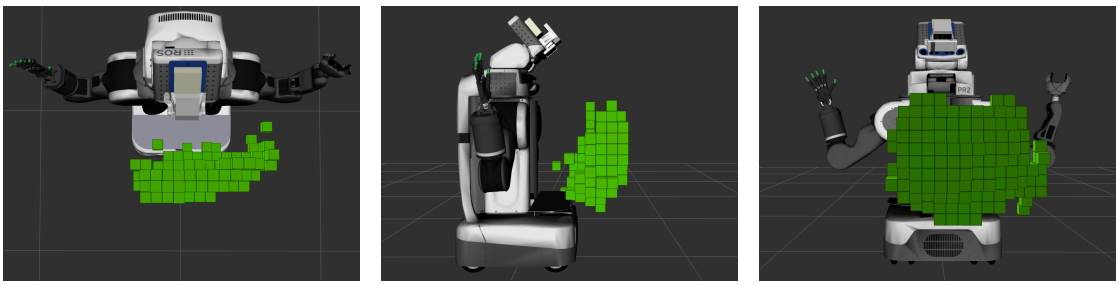


Figure A.2.: Additional views of the sliced version of all valid solutions for the side grasp.



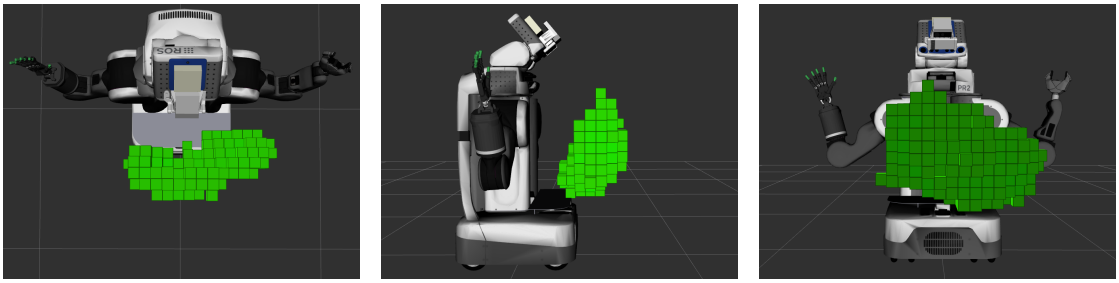
(a) Top (b) Side (c) Front

Figure A.3.: Additional views of the best 5% regarding the number of valid solutions for the side grasp.



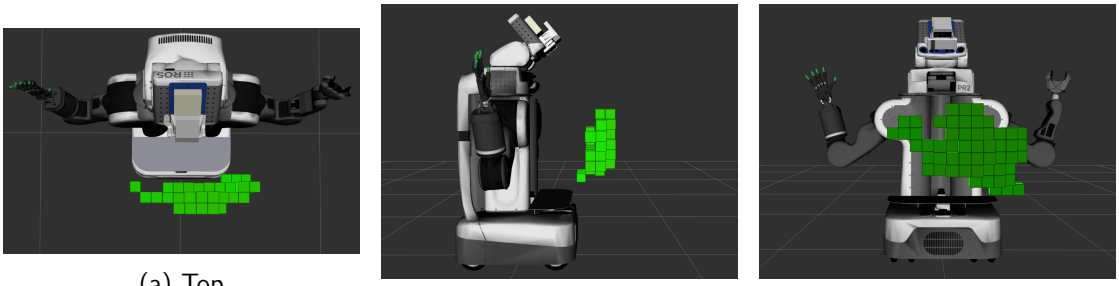
(a) Top (b) Side (c) Front

Figure A.4.: Additional views of the best 5% regarding the average cost for the side grasp.



(a) Top (b) Side (c) Front

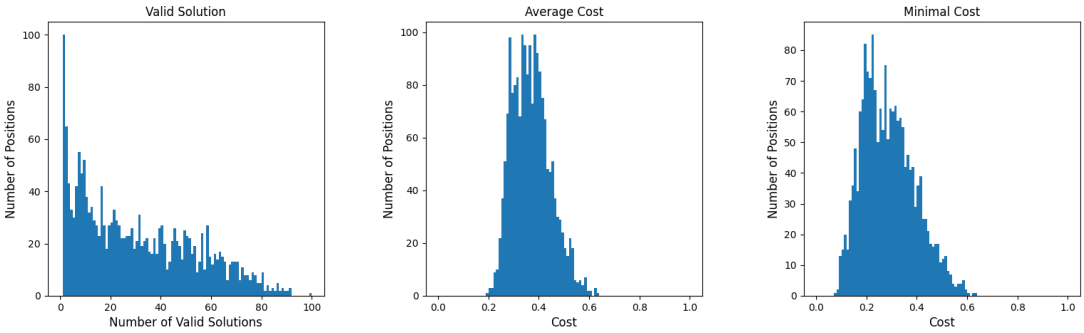
Figure A.5.: Additional views of the best 5% regarding the minimal cost for the side grasp.



(a) Top (b) Side (c) Front

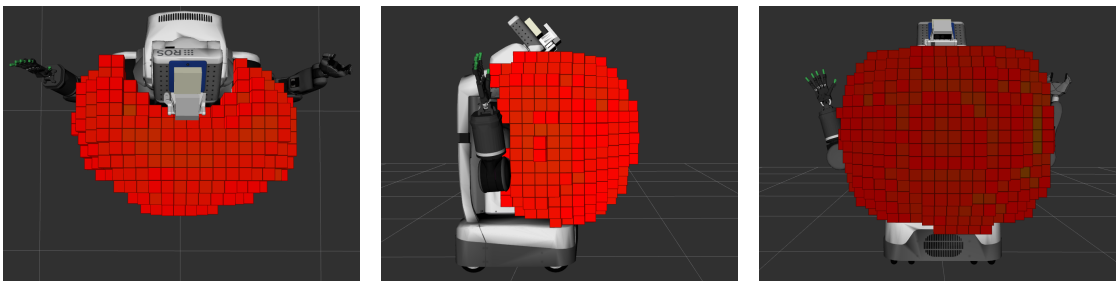
Figure A.6.: Additional views of the intersection of the best 5% regarding all metrics for the side grasp.

A.2. Side Grasp X-Shift



(a) Number Solutions (b) Average Cost (c) Minimal Cost

Figure A.7.: Distribution of all valid solutions for the x-shifted side grasp according to the three defined metrics of number of valid solutions, average cost and minimal cost.



(a) Top (b) Side (c) Front

Figure A.8.: Additional views of all valid solutions for the x-shifted side grasp.

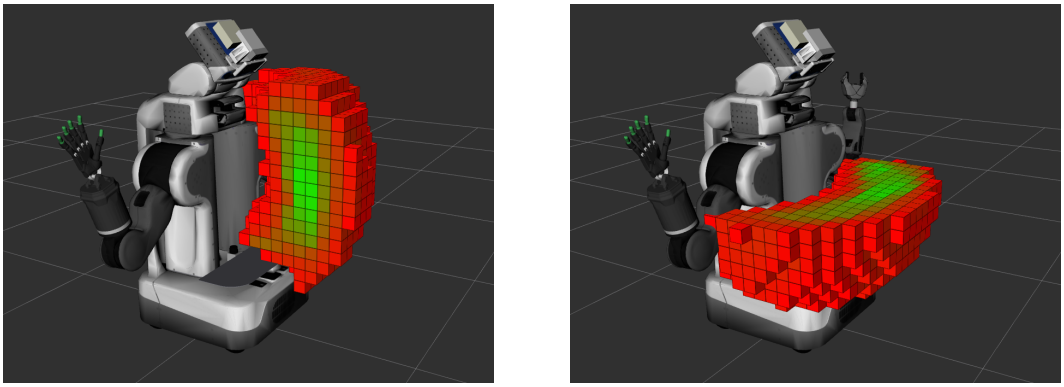
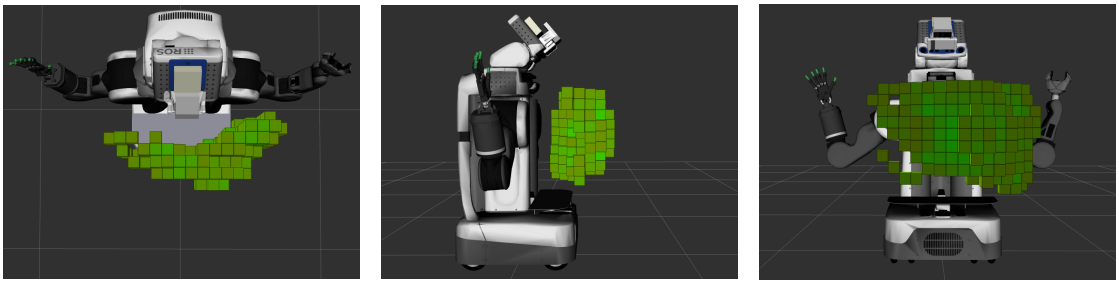
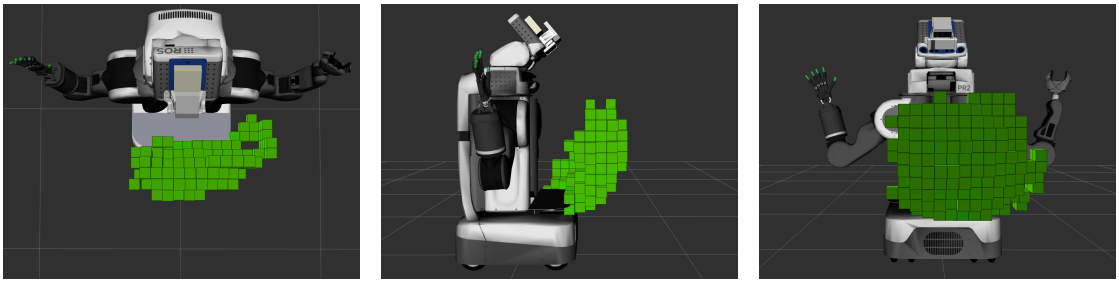


Figure A.9.: Additional views of the sliced version of all valid solutions for the x-shifted side grasp.



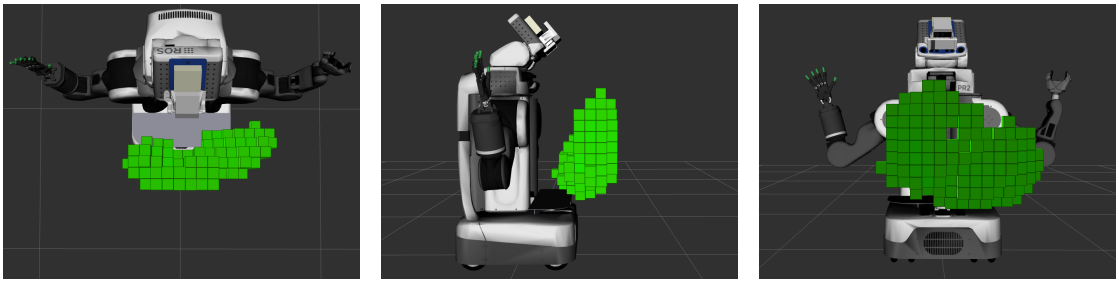
(a) Top (b) Side (c) Front

Figure A.10.: Additional views of the best 5% regarding the number of valid solutions for the x-shifted side grasp.



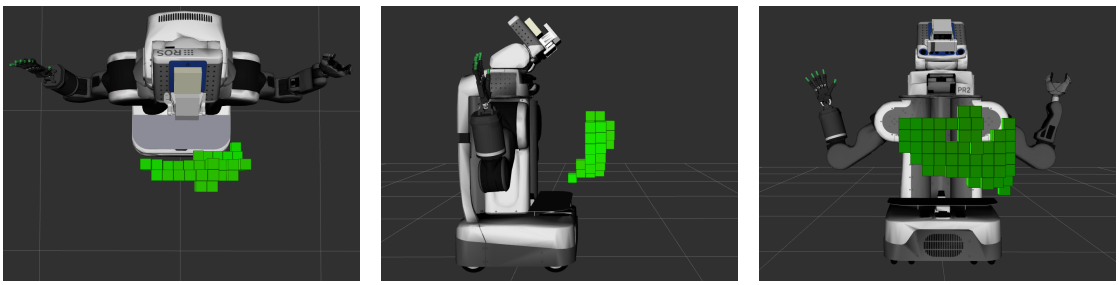
(a) Top (b) Side (c) Front

Figure A.11.: Additional views of the best 5% regarding the average cost for the x-shifted side grasp.



(a) Top (b) Side (c) Front

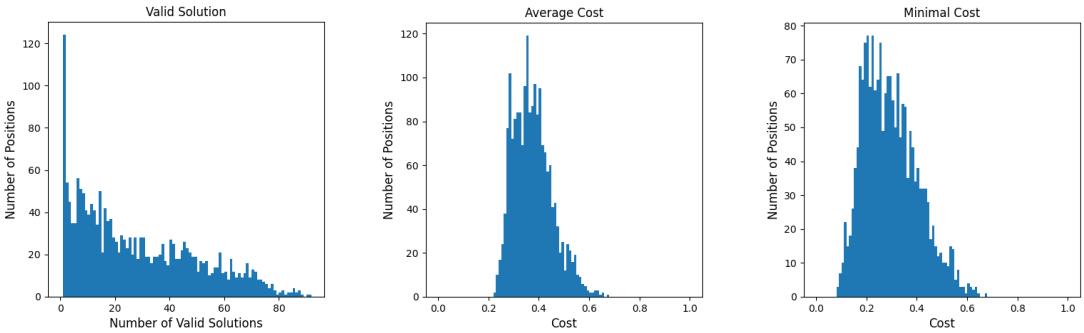
Figure A.12.: Additional views of the best 5% regarding the minimal cost for the x-shifted side grasp.



(a) Top (b) Side (c) Front

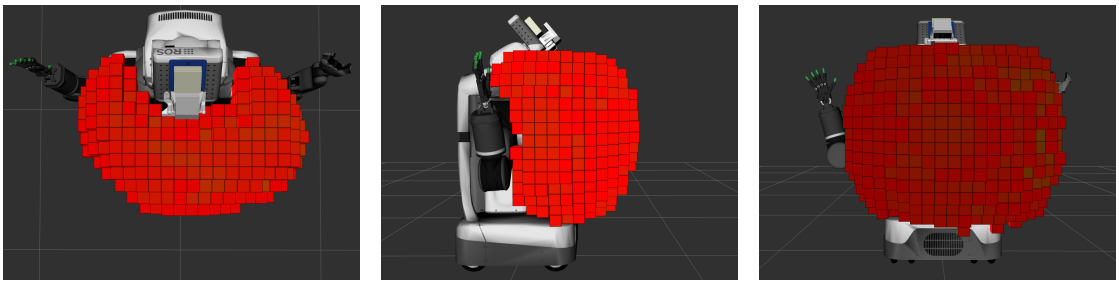
Figure A.13.: Additional views of the intersection of the best 5% regarding all metrics for the x-shifted side grasp.

A.3. Side Grasp Y-Shift



(a) Number Solutions (b) Average Cost (c) Minimal Cost

Figure A.14.: Distribution of all valid solutions for the y-shifted side grasp according to the three defined metrics of number of valid solutions, average cost and minimal cost.



(a) Top (b) Side (c) Front

Figure A.15.: Additional views of all valid solutions for the y-shifted side grasp.

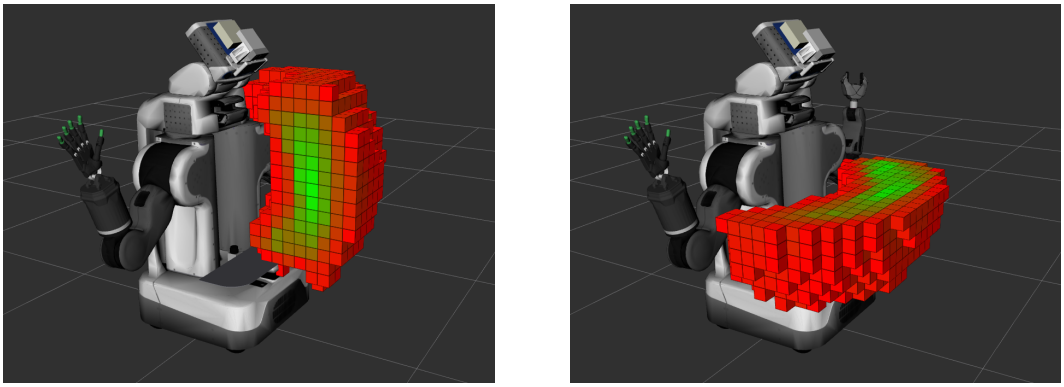
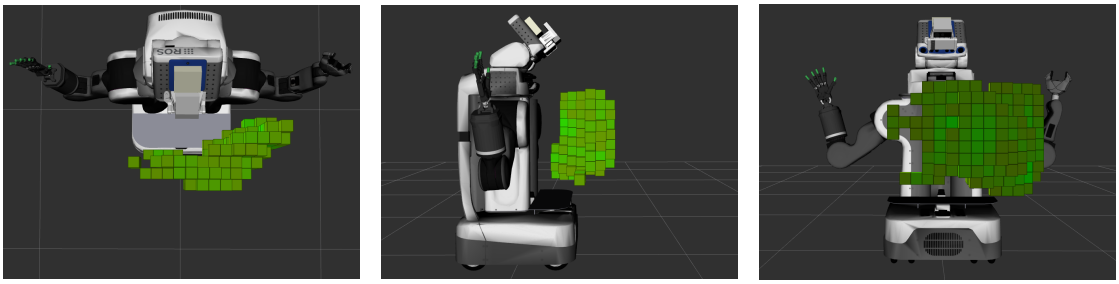
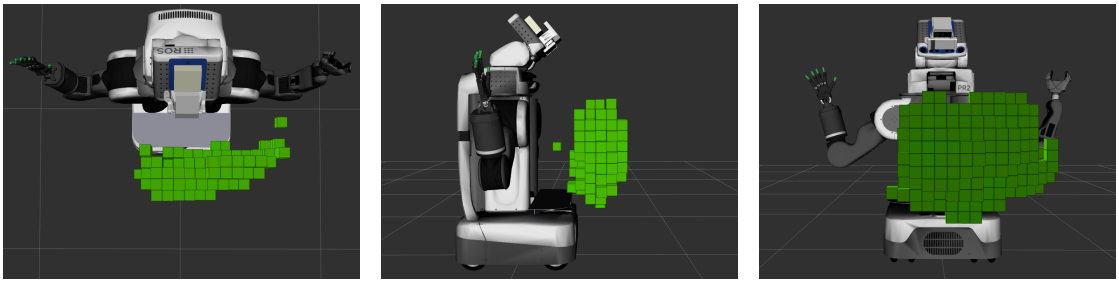


Figure A.16.: Additional views of the sliced version of all valid solutions for the y-shifted side grasp.



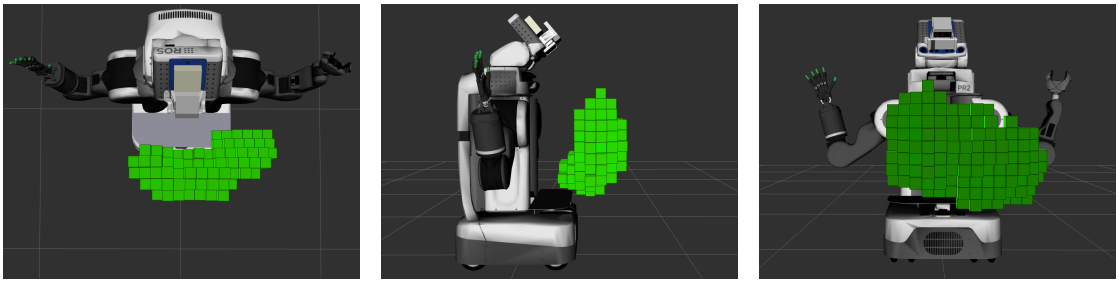
(a) Top (b) Side (c) Front

Figure A.17.: Additional views of the best 5% regarding the number of valid solutions for the y-shifted side grasp.



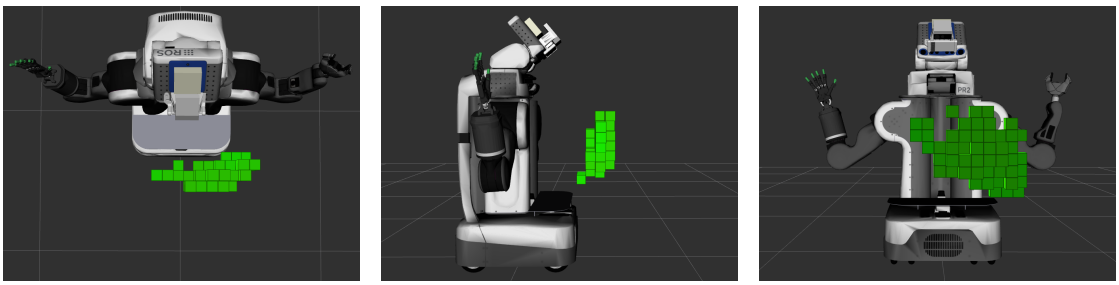
(a) Top (b) Side (c) Front

Figure A.18.: Additional views of the best 5% regarding the average cost for the y-shifted side grasp.



(a) Top (b) Side (c) Front

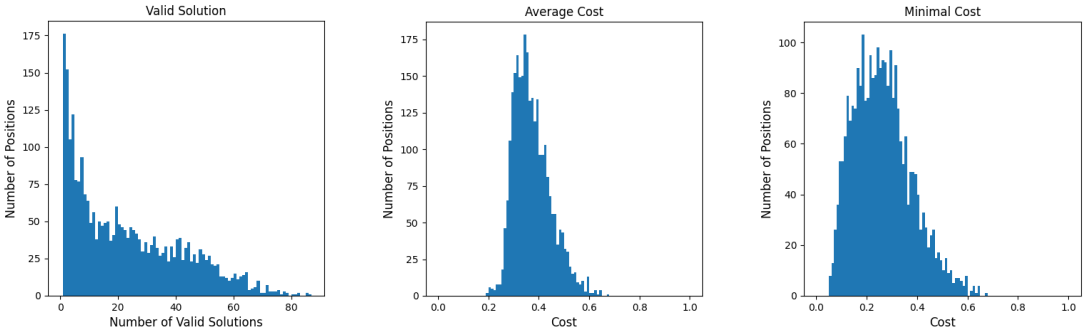
Figure A.19.: Additional views of the best 5% regarding the minimal cost for the y-shifted side grasp.



(a) Top (b) Side (c) Front

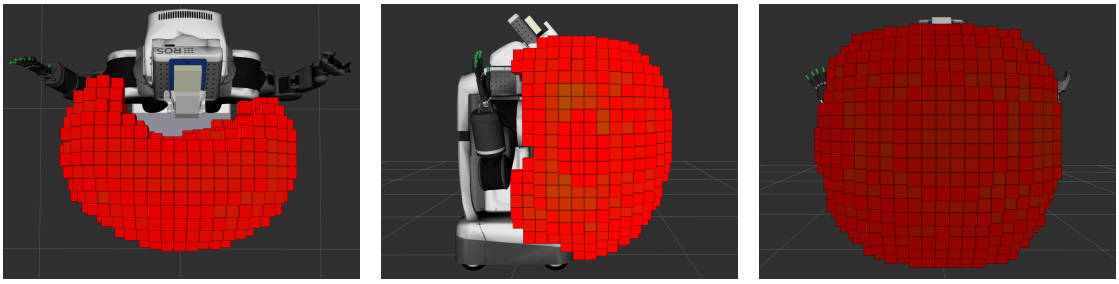
Figure A.20.: Additional views of the intersection of the best 5% regarding all metrics for the y-shifted side grasp.

A.4. Top Grasp



(a) Number Solutions (b) Average Cost (c) Minimal Cost

Figure A.21.: Distribution of all valid solutions for the top grasp according to the three defined metrics of number of valid solutions, average cost and minimal cost.



(a) Top (b) Side (c) Front

Figure A.22.: Additional views of all valid solutions for the top grasp.

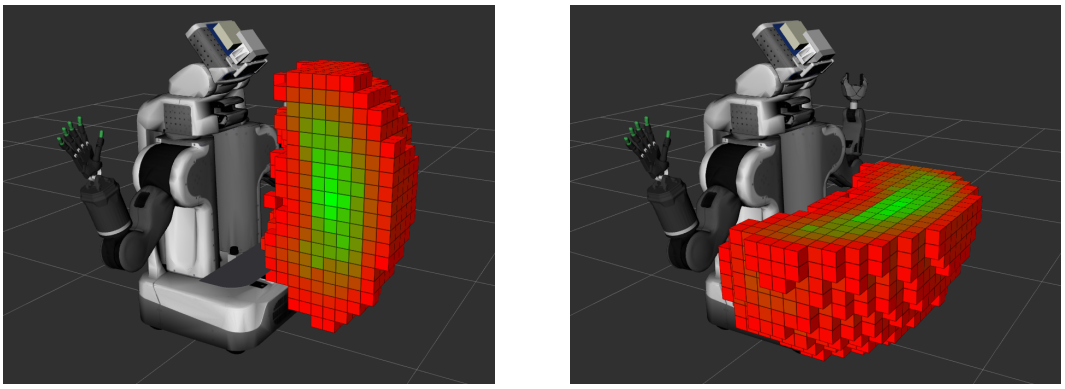
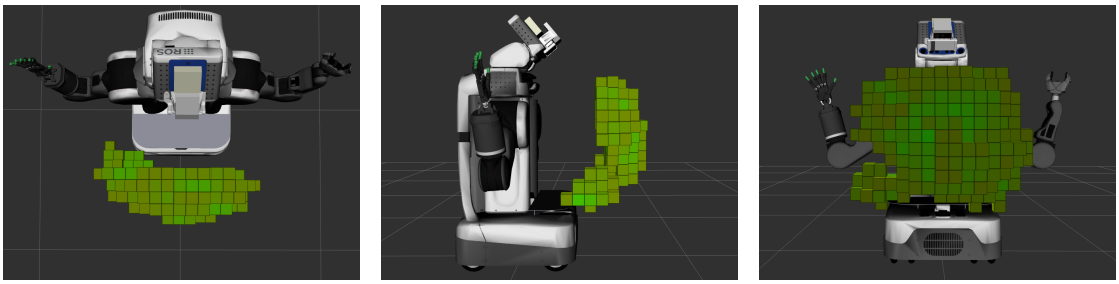
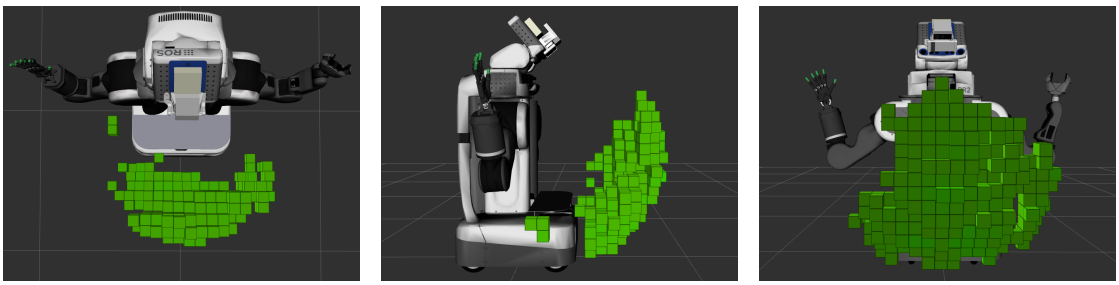


Figure A.23.: Additional views of the sliced version of all valid solutions for the top grasp.



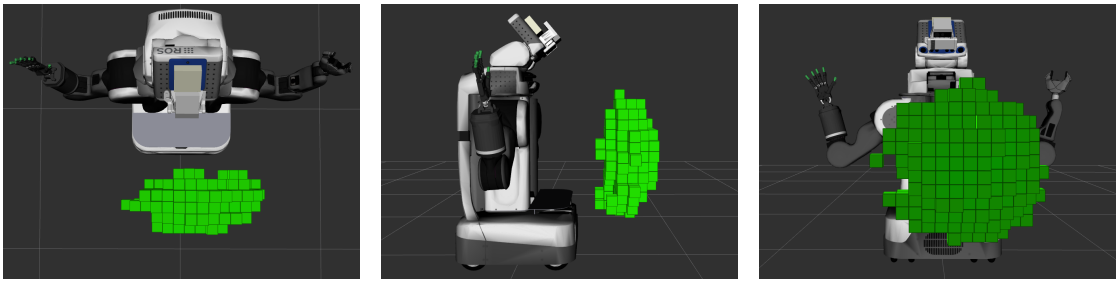
(a) Top (b) Side (c) Front

Figure A.24.: Additional views of the best 5% regarding the number of valid solutions for the top grasp.



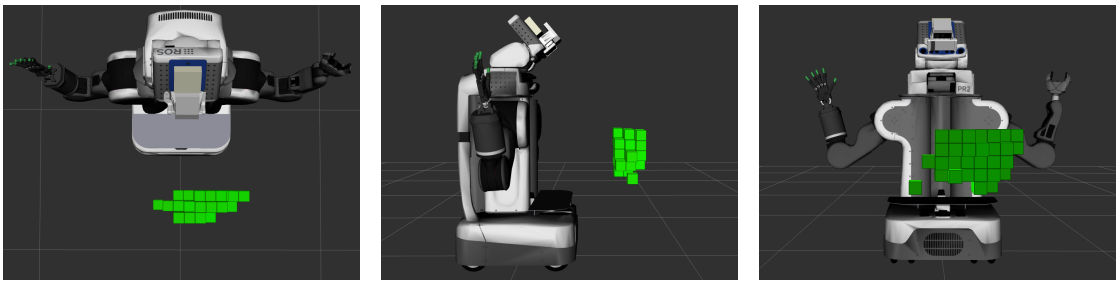
(a) Top (b) Side (c) Front

Figure A.25.: Additional views of the best 5% regarding the average cost for the top grasp.



(a) Top (b) Side (c) Front

Figure A.26.: Additional views of the best 5% regarding the minimal cost for the top grasp.



(a) Top (b) Side (c) Front

Figure A.27.: Additional views of the intersection of the best 5% regarding all metrics for the top grasp.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel — insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen — benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe.

Hamburg, den 30.03.2024

Björn Sygo

Björn Sygo

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den 30.03.2024

Björn Sygo

Björn Sygo

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel — insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen — benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe.

Hamburg, den 30.03.2024

Björn Sygo

Björn Sygo

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den 30.03.2024

Björn Sygo

Björn Sygo