End-to-end behavior cloning agent for an object handover task

Training and evaluating a robot to perform simulated human-to-robot object handovers without requiring hand-object segmentation

MSc Thesis

Yuki Watabe

TU Delft

End-to-end behavior cloning agent for an object handover task

Training and evaluating a robot to perform simulated human-to-robot object handovers without requiring hand-object segmentation

by

Yuki Watabe

Student Name	Student Number
Yuki Watabe	4669797

Instructor:	Dr.ir. Yke Bauke Eisma
Daily Supervisor:	Renchi Zhang
Committee Members:	Dr. ir. Yke Bauke Eisma
	Dr. Dimitra Dodou
	Renchi Zhang
Faculty:	Faculty of Mechanical Engineering, Delft

Cover: HandoverSim
Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Contents

Abstract

Current visuomotor manipulators jointly train their perception-planning-action components simultaneously using an end-to-end framework to avoid hand-engineering components. Despite this, methods for human-to-robot object handover tasks require a perception component that segments the hand from the object, which can introduce error propagation. For this reason, this study investigates the applicability of an end-to-end framework that eliminates the need for hand-object segmentation in a simulated human-to-robot object handover task using HandoverSim.

To address this, a behavior cloning agent is used to convert camera input into RGB-D voxel space and output discretized 6-DoF manipulation to directly discover features for the handover task. This study introduces a framework that combines the behavior cloning agent with the HandoverSim, which allows experimenting with various training configurations. These configurations consist of experiments with: 1) expert demonstration data; 2) optimal camera setup; 3) handover objects; and 4) voxel-based RGB augmentation techniques.

The trained model is evaluated on its generalization to diverse handover conditions in the HandoverSim Benchmark. The results demonstrate that the behavior cloning agent can learn features for the handover task without requiring a perception component. The model learns the grasp-object relation whilst minimizing contact with the hand. Despite this, performance is limited by sparse training data and grasping accuracy.

1. Introduction

1.1. Background

Current studies show an increased interest in jointly training the perception and control systems in an end-to-end manner for visual-motor (visuomotor) robotic manipulators. Contrary to pipelined approaches, this framework trains the perception-planning-action components simultaneously using deep neural networks and infers motor actions directly from (depth) cameras. Ultimately, this allows end-to-end approaches to avoid hand-engineering beforehand and prevents error propagation. [1]
In the visuomotor domain, grasping is well-researched for requiring dexterity in perception, planning, and control, and relies on data-driven approaches that leverage grasping datasets to sample grasps and rank them according to a grasping metric [2] [3]. Prior methods assume planar grasping by restricting a perpendicular view to the grasping scene. This grasping method has shown success in increasingly complex environments (structured, cluttered, occluded, and novel objects) and has been established using benchmarks with varying degrees of complexities [4] [5] [6] [7] [8] [9]. However, planar grasping limits 3D reasoning by neglecting a large number of possible grasps, and applications by not exploiting the full kinematic capabilities of the robot [10]. In contrast, 6-DoF grasping addresses these limitations by incorporating depth information and has shown similar advancements in increasing complex environments [10] [11] [12]. Additionally, learning affordance-based actions, which involve interpreting potential actions based on the object and its location, enhances grasping to predict a range of possible actions at regions that are likely to exhibit movement [13] [14] [15] [16].

1.2. Object handover

Current advancements of grasping methods have proven effective through standardized benchmark environments [3]. Yet, research on collaborative tasks involving human interaction remains limited. In addition to safety concerns when interacting with a robot, end-to-end visuomotor frameworks require large amounts of data that are difficult to gather because of the lengthy and costly sampling time when learning with a human-in-the-loop [17].
In contrast, object handovers have shown an increased focus in recent years by adapting previously mentioned grasping methods to work by segmenting the hand from the grasped object [18] [19]. This collaborative task is a joint action between a giver and a receiver, whereby joint actions are defined as [20]:

> any form of social interaction whereby two or more individuals coordinate their actions in space and time to bring about a change in the environment ... successful joint action depends on the abilities (i) to share representations, (ii) to predict actions, and (iii) to integrate predicted effects of own and others' actions.

To accomplish the handover, the giver must present the object to the partner, maintain a stable grasp during the physical handover, and finally release the object to the receiver as safely as possible. Conversely, the receiver must acquire the object by grasping, stabilize its grasp on the object, and potentially proceed to use the object for its intended purpose [21].
Furthermore, a handover can be structured into a pre-handover phase and a physical exchange phase as illustrated by Figure 1.1. The pre-handover phase contains communication between agents for grasping and transferring the object by the giver. The receiver detects and tracks changes in the surrounding dynamic environment, and must predict and adapt its motion planning and control based on

the giver's intentions. The physical handover is initiated by the contact of the receiver's hand with the object and ends when the giver removes their hand from the object and is fully held by the receiver [21].

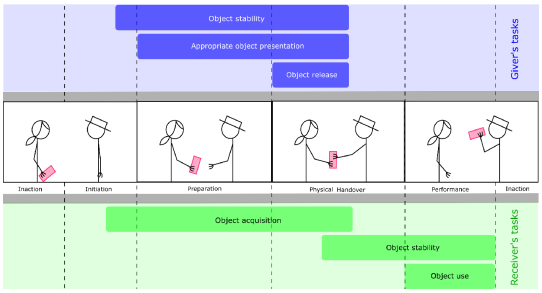


Figure 1.1: Handover overview with task description of giver vs. receiver. [21]

Specifically, in this research, the giver is the human and the receiver is the robot. The receiver aims to: (1) acquire the object by grasping; (2) stabilize the grasp on the object; and finally, following the handover, (3) perform the task the object was required for.

Given the challenges associated with human-in-the-loop data collection, imitation learning presents a promising alternative to reduce the costly process of data collection by relying on expert demonstrations to learn tasks [1] [22] [23]. The most popular method for imitation learning is behavior cloning (BC), which reduces imitation learning to a standard supervised learning problem by requiring only trajectories and makes them widely applicable [24].
In addition, a simulation for human-robot handover, HandoverSim [25], has been created to achieve benchmarking handovers in which the human acts as the giver and the receiver as the robot [25]. This simulation allows for benchmarking with a standardized and reproducible evaluation of receiver policies. The benchmark has been evaluated on a motion planning method [26], a task planning method [27], and a pre-trained reinforcement learning policy [19]. The first two methods evaluate with ground-truth state inputs and utilize pre-generated grasp poses from the works of Eppner et al. [28] for each object in the HandoverSim. The last method uses a pre-trained reinforcement learning policy, GA-DDPG [19], that has been trained in a tabletop grasping setting that is comparable to HandoverSim and uses the same PyBullet Physics engine [29]. Notably, none of the methods have been trained using HandoverSim and do not explore policy training with human-in-the-loop. For this reason, Christen et al. [30] trained their policy within the handover benchmark to provide simulated human-in-the-loop training using HandoverSim to react to dynamic human behavior. Despite this, previously mentioned methods [19] [30] evaluated in HandoverSim require a perception module that segments the hand from the object (hand-object segmentation). While the hand-object segmentation mask was retrieved from ground truth in the HandoverSim, these methods suffer from errors introduced in the perception pipeline by Yang et al. [27] when evaluating sim-to-real transfer.

1.3. Research proposal

To address these gaps, this study investigates the applicability of a fully end-to-end framework that eliminates the need for hand-object segmentation for the context of a simulated human-to-robot object handover task using the HandoverSim. In particular, a BC agent was leveraged to convert camera input into RGB-D voxel space and output discretized 6-DoF manipulation to directly discover features for the handover task. The BC agent was trained using expert demonstration collected from one of the baselines evaluated in Handoversim. To facilitate BC agent training, data provision methods from the expert demonstrations were looked into. Additionally, the optimal camera setup was analyzed to cover the task space. Furthermore, a voxel space augmentation technique was analyzed to create a robust model similar to image augmentation for computer vision. Finally, the trained model is evaluated on its generalization to various handover conditions from the handover benchmark.
To recapitulate, the research question is summarized as: **Assessing an end-to-end behavior cloning agent for discretized 6-DoF manipulation via RGB voxel inputs in a human-to-robot object handover benchmark**.

2. Methodology

This chapter is divided into two sections. The first section introduces the tools that were used throughout this research. This section provides justification for the use of the methods, a description of the method, and inputs required for the use of these methods. The second section introduces the framework to train and evaluate the end-to-end BC agent for the handover task. Moreover, potential experimental variations for the training and evaluation configuration are introduced, which are materialized in chapter 3.

2.1. Selected Tools

This section describes the tools used for running a BC agent on a human-to-robot object handover. Specifically, a behavior cloning agent, Perceivor-Actor [31], is used in combination with a simulation that can benchmark human-to-robot object handover, HandoverSim [25].

2.1.1. Perceiver-Actor

For imitation learning, a behavior cloning agent was used by Perceiver-Actor (PerAct) [31]. PerAct is a language-conditioned behavior cloning agent that directly learns to detect actions from a unified raw input observation. This prevents creating perception modules, such as object detectors, instance-segmenters, or pose-estimators, to produce a task-specific scene for learning a policy and makes PerAct applicable for a broad range of tasks [31].

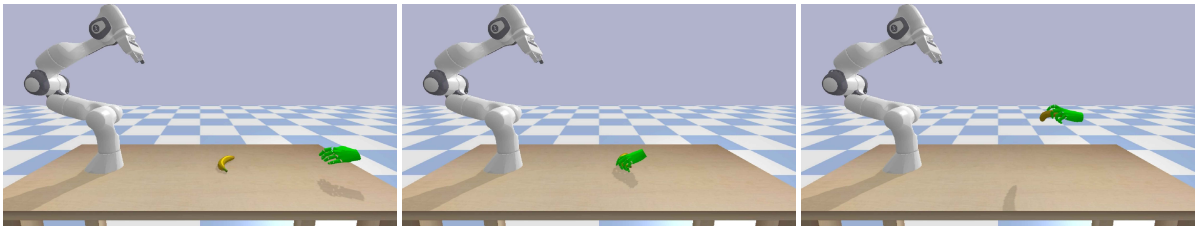
For using PerAct, the raw input observation contains {*a language goal* and *RGB-D camera images*}. The RGB-D camera observations are fused to construct a voxel grid from known camera intrinsics and extrinsics. The voxel grid and language goal are fed into a PerceiverIO transformer [32] to encode into per-voxel features. The per-voxel features are used to output Q-value actions by their respective translation, rotation, and gripper-open components. The translation Q-value is given per voxel in its original voxel grid, rotation is represented by a rotation bin per axis (roll, pitch, yaw), and gripper-open is a binary value. The best predicted (discretized) action is chosen by taking the maximum value, which can be executed using a motion planner.

To train PerAct, the model is supervised by expert demonstrations that contain a sequence of actions to perform a certain task. Per demonstration, the end-effector must go through bottleneck action(s), whereby each bottleneck action is referred by a keyframe action [33]. PerAct utilizes a keyframe extraction method by using a heuristic when (1) the joint-velocities are near zero and (2) the gripper open state has not changed. The keyframe action is discretized in the same manner as the aforementioned PerceiverIO discretized output. Hence, the goal is to predict the next best action given a voxel observation, which is similar to a classification task but in 3D-voxel space. The loss function for training the model is formulated by a cross-entropy loss like a classifier and is given in Equation 2.1. Each term in Equation 2.1 calculates the cross-entropy loss of the softmax Q-values in translation, rotation (accumulation of roll, pitch, yaw), and gripper-open state, respectively. The PerAct loss contains a collision term in addition to Equation 2.1, but this term is ignored since the end-effector must get in contact with the (object's) voxels.

$$\mathcal{L}_{total} = -\mathbb{E}_{Y_{trans}}[\log \nu_{trans}] - \mathbb{E}_{Y_{rot}}[\log \nu_{rot}] - \mathbb{E}_{Y_{open}}[\log \nu_{open}] \tag{2.1}$$

2.1.2. HandoverSim benchmark

For collecting demonstrations and evaluating PerAct for a handover task, a human-to-robot handover simulation benchmark, HandoverSim [25], was used. This benchmark captures human-to-robot object handovers and evaluates the receiver's (robot's) policy for the capability of grasping the objects. This simulation benchmark simulates the giver's motion by leveraging a motion capture dataset for hand-grasping objects, DexYCB [34]. DexYCB has recorded human subjects picking a target object from a table of objects and holding the object in the air to pretend handing over an object to someone across from them. For each handover event, the 3D pose of both the hand and the objects is used for reproducing the human giver's motion for the pre-handover phase in the simulation. An example of the handover sequence is illustrated in Figure 2.1.



(a) Human reaches to the object. **(b)** Human grasps the object. **(c)** Human presents the object to the robot.

(d) Robot moves towards the object. **(e)** Robot grasps the object. **(f)** Robot moves to goal with grasped object.

Figure 2.1: Human (giver) to robot (receiver) object (here: banana) handover simulated in 6 stages. The top row contains actions from the giver, which have been pre-generated using DexYCB [34]. The bottom row contains inferred actions from the receiver's policy.

DexYCB has recorded 10 subjects grasping 20 objects with 5 trials per subject-object pair, which totals 1000 handover sequences. However, two objects are omitted in HandoverSim, since not all objects in DexYCB are graspable by the parallel-jaw gripper of the Franka Emika Panda robot in the simulation. Since the HandoverSim Benchmark is not only used for evaluating, Chao et al. [25] separate all captured trials into train, validation, and test sets. The handover sequences are divided into four setups that categorize a handover. A summary is given below and the distribution of the sequences in Table 2.1.

- S0 (default): The train split contains all 10 subjects and all 18 grasped objects
- S1 (unseen subjects): The scenes are split by human subjects (train/val/test: 7/1/2).
- S2 (unseen handedness): The scenes are split by the left or right hand used (train/val/test: right-hand/left-hand/left-hand).
- S3 (unseen grasping): The scenes are split by the handed-over object (train/val/test: 14/2/2).

Table 2.1: Statistics for the S0 (default), S1 (unseen subjects), S2 (unseen handedness), and S3 (unseen grasping) setup of HandoverSim [25]. For this research, S1 was chosen for training and evaluation.

	S0: default				**S1: unseen subjects**				S2: unseen handedness				S3: unseen grasping			
	#sub	hand	#obj	#sce	**#sub**	**hand**	**#obj**	**#sce**	#sub	hand	#obj	#sce	#sub	hand	#obj	#sce
train	10	R/L	18	720	**7**	**R/L**	**18**	**630**	10	R	18	449	10	R/L	14	700
val	2	R/L	18	36	**1**	**R/L**	**18**	**90**	2	L	18	91	10	R/L	2	100
test	8	R/L	18	144	**2**	**R/L**	**18**	**180**	8	L	18	360	10	R/L	2	100
all	10	R/L	18	900	**10**	**R/L**	**18**	**900**	10	R/L	18	900	10	R/L	18	900
all*	10	R/L	20	1,000	10	R/L	20	1,000	10	R/L	20	1,000	10	R/L	20	1,000

* "all*" includes all sequences from DexYCB [34], and "all" is after removing objects ungraspable by the gripper.

Each handover in the benchmark is defined as *success* when the following conditions are both met for 0.1 seconds:

1. The gripper fingers are in contact with the handed-over object.
2. The position of the gripper link lies with a pre-specified goal region.

Therefore, to clarify the aim of the receiver described by Ortenzi et al. [21] (i.e. *(3) perform the task the object was required for*), the handover task is completed when the grasped object is moved to the goal region as illustrated by Figure 2.1f with above conditions.

Contrarily, a *failure* is detected when either one of the following three conditions is met:

- *Collision*: Any part of the robot body is in contact with any part of the human hand.
- *Dropped object*: At least one of the gripper fingers is not in contact with the handed over object and the object is in contact with the table or other objects or its center falls below the height of the table surface.
- *Timeout*: A maximum time limit of 7 seconds is reached[1].

2.2. End-to-end behavior cloning framework for object handovers

Figure 2.2 provides a structured overview of the workflow in this research and shows how HandoverSim and PerAct are integrated. In the attempt to achieve a successful object handover using an end-to-end BC agent, the following subsections introduce training and evaluation configurations. Specifically, 1) data acquisition and utilization methods for supervising PerAct with expert demonstration (concerning the gray box and orange cylinder), 2) the optimal camera setup for task-scene coverage (concerning the gray and red box) 3) properties of handover objects (concerning the gray and red box), and 4) looking into data augmentation to create a robust policy (concerning the green box). The experiments are concretely defined in chapter 3. Each configuration element introduced per subsection is an integral part of the workflow pipeline and allows for either improving or evaluating the capabilities of PerAct in the handover task.

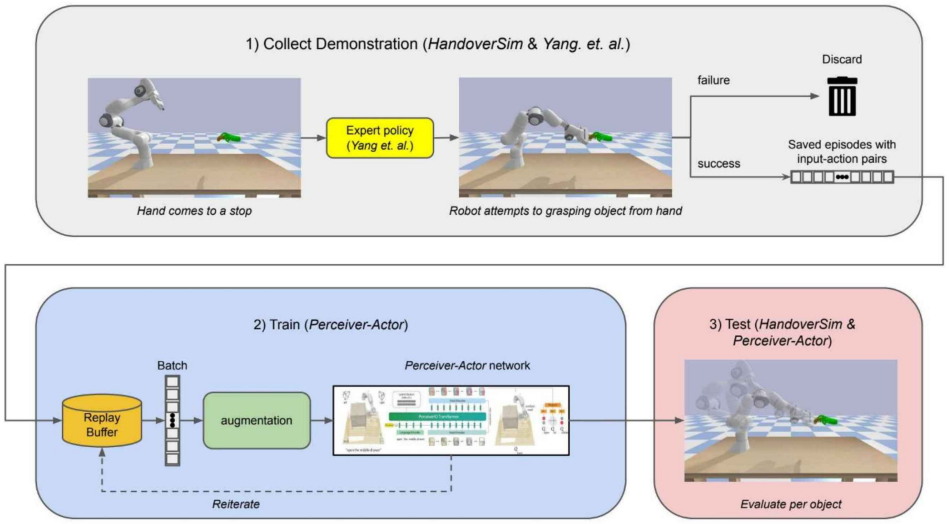


Figure 2.2: Framework for training and evaluation with HandoverSim [25] and Perceiver-Actor [31]. 1) During expert demonstration collection (top/gray box), an expert policy by Yang et al. [27] attempts to grasp the object from the hand, which is saved if successful. 2) During training (bottom-left/blue box), a batch of sampled transitions from the replay buffer is passed through the Perceiver-Actor network for training. Here, samples can be augmented before passing through the training network. 3) For evaluation (bottom-right/red box), the trained behavior cloning agent interacts in the HandoverSim environment.

[1]The paper uses 13 seconds.

Among the four setups introduced in subsection 2.1.2, the S1 setup (unseen subjects) was used in this research for collecting data and evaluating PerAct on the handover task. This setup offers the highest diversity of grasping styles across each split and can evaluate the robot's ability to handle handovers with subjects it has never encountered before [34]. In contrast, S0 and S1 may contain some handover style bias, since subjects' styles at evaluation have been seen during training, which could lead to bias. Similar to the works of Shridhar et al. [31], the generalization to unseen objects is not evaluated, which omits setup S3. In contrast to methods evaluated in HandoverSim, the PerAct agent is trained per object only.

Furthermore, the HandoverSim Benchmark was modified to capture the handover sequence in ARM codebase storage format [35], which can be directly used with PerAct. The codebase for this can be found in this GitHub repository[2].

2.2.1. Expert demonstration: keyframe extraction and sampling

While the HandoverSim simulation benchmark does provide a training and evaluation setup, the benchmark does not contain a ground-truth method for grasping objects. The policy from Yang et al. [27] was used to obtain expert demonstrations and is illustrated in Figure 2.2 by the yellow box. This baseline has also been evaluated to benchmark in the HandoverSim and yields the highest success on the S1 setup. Although the policy works by generating grasps from tracking and segmenting the object from the hand, the implementation of Yang et al. in the benchmark bypasses those grasp generations by using pre-generated grasp poses that have already been registered for the handover objects [28]. Each successful attempt by Yang et al. is stored as an expert demonstration, whilst failure is disregarded. This is illustrated by the two arrows in the right section of the top/gray box in Figure 2.2.

Given the handover task, a keyframe action involves immediately grasping the object without collision with the hand. However, since actions are executed using a motion planner, moving directly to this action could result in a collision with the hand or object during its path. In this analysis, the idea is to look into collecting multiple keyframes to safely receive the object from the hand. Taking inspiration from the works of Yang et al. [27] and Christen et al. [30], each handover can be safely approached through a two-staged attempt. At its first stage, the end-effector approaches the object at a safe distance while orienting itself for the grasping pose. Finally, because the orientation is already collision-free, the end-effector only has to reach forward to grasp the object. Hence, the bottleneck action is to go through this approach phase for each handover task.

The collected expert demonstrations from the HandoverSim are stored in a replay buffer (illustrated by the orange cylinder in Figure 2.2) with input-action pairs. Each input-action sample is randomly replayed to supervise PerAct by predicting the corresponding keyframe action based on the given observation. Concerning the two-stage actions (approach & grasp) for performing a handover, various sampling methods are analyzed to discover a sampling method that is best suited for PerAct to learn the handover task. Furthermore, possible data imbalance introduced by over- or under-sampling of certain keyframe actions is addressed by comparing these sampling methods and is examined by analyzing the learned behavior.

2.2.2. Multi-camera setup

To provide expert demonstrations to PerAct (gray box in Figure 2.2), RGB-D camera images must be captured that cover the task space using any number of cameras. The RGB-D images are converted to voxels to form a volumetric representation of the task scene. This transforms the 1m x 1m x 1m task space into a voxel space of 100 x 100 x 100 cells. The voxelized 3D observations will provide a strong structural prior for learning 6-DoF actions [31]. The works of Jauhri et al. [36] have already observed that capturing the depth from certain views can yield poorly perceived reconstructions of the task scene. In their experiment, thin objects are indistinguishable from the table, leading to a grasp failure. Even though the handover task occurs above the table, the quality of voxel observations could still influence the handover task. For this reason, various camera configurations are analyzed regarding the voxel quality of the handover scene, and their influence on the handover task is evaluated.

The multi-camera configuration analyzes camera numbers and placements to cover the handover

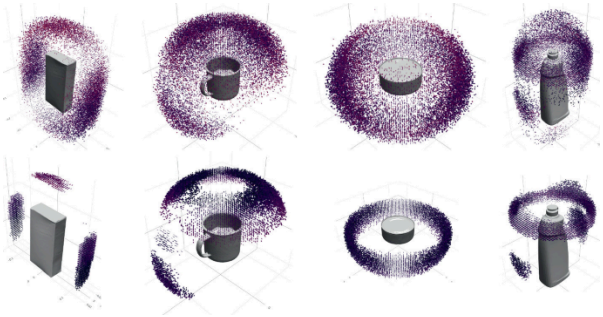
[2] https://github.com/yuki1003/handover-sim

scene or handover object. The expected result is that a high representation of 3D voxel observation will lead to higher success and precise grasps.

2.2.3. Handover Objects

HandoverSim provides 18 distinct YCB objects [37] available for object handovers, as provided in Figure 2.3a. While the simulation is designed for training using a mixture of objects, this research individually evaluates objects. Although the focus is on a single-task agent's capability for object handovers, the agent can be evaluated per YCB object to identify the relation of object properties with PerAct's handover capabilities. Object properties such as shape, texture, and size can limit the grasping ability or produce various combinations of grasps. Figure 2.3b gives an example of available (robust) grasps that are affected by shape [28]. This method consists of collecting object properties, analyzing the failure types that occur for each handover object, and finding a relation between the failure type and with object property.

(a) YCB objects available in HandoverSim, excluding the *large marker*, *master chef can*, and *wood block*. [38]



(b) Examples of available grasps for YCB objects. The bottom row shows robust versions of the grasp set. [28]

Figure 2.3: Available grasps for YCB objects in the handoversim.

2.2.4. Voxel-based RGB augmentations

Deep neural networks, such as the end-to-end frameworks for visuomotor policies, require a lot of data to obtain good results and prevent overfitting [1]. Data randomization techniques, such as from the works of Tobin et al. [39], apply variability in lighting, textures, object sizes, camera angles, etc., to obtain a model that shows robustness against distractors, lighting conditions, changes in the scene, and moving objects.

PerAct uses data augmentation at training by enabling each voxel observation to be augmented with SE(3) transformation, which yields a perturbation in translation and rotation. The translation perturbation has a range of $\pm$ 0.125 meters along each axis and a rotation of $\pm$ 45 $^\circ$ along the yaw axis. This augmentation has been shown to improve performance on tasks with rotation variation. Furthermore, their method has been evaluated on the generalization to novel instances and objects, by perturbing object properties of the `open drawer` task: color, texture, scale, and gripper shape. However, these perturbations are not applied during training. While indicating robustness to gripper shape, other object property changes yield confusion. It is suspected that training on several instances of drawers might improve generalization performance. Considering the handover task, this analysis looks into training with RGB augmentations (akin to the work of Tobin et al. [39]) to create a robust handover model. The voxel augmentations are illustrated by the green box in Figure 2.2 and are applied on the batch samples prior to inputting the PerAct network.

3. Experiments

In this chapter, the aforementioned methods from section 2.2 are more concretely defined by their experimental setup. The data acquisition and usage from the expert demonstration is defined in section 3.1. Firstly, it introduces the keyframe actions acquired for the handover task. Secondly, various sampling methods are described for the input-action pairs. Furthermore, section 3.2 considers various camera setups for capturing the handover task scene. Hereafter, section 3.3 describes the collected demonstrations for each object present in the HandoverSim. Lastly, section 3.4 describes the implementation of the voxel-RGB augmentation. In chapter 4, the results for each experiment are provided.

3.1. Expert demonstration: keyframe extraction and sampling

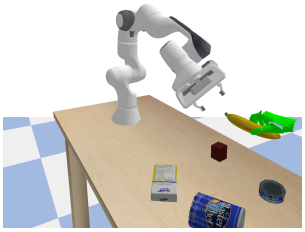
As observed in subsection 2.2.1, the expert demonstration will be sampled as input-action pairs during training. In the following subsections, the method to obtain keyframe actions is introduced. Hereafter, various sampling methods are introduced that consider the relation of acquired keyframe actions with input observations.

3.1.1. Keyframe extraction

(a) Gripper approaches handle.

(b) Gripper grasps the handle.

(c) Open the drawer.



(d) Gripper approaches object.

(e) Gripper grasps the object.

(f) Gripper moves to the goal.

Figure 3.1: Keypoints for the `open drawer` task from PerAct that is analogous to the handover task.

Applying the heuristic from PerAct to the `open drawer` task that is provided as an example by the PerAct GitHub repository yields the following keyframes that depict: 1) an end-effector finding an appropriate

orientation that can yield to grasping the handle perpendicular to the drawer's surface, 2) approaching the handle of the drawer and closing its gripper, and 3) retracting whilst keeping its gripper closed to open the drawer. The keyframes provided by the example `open drawer` task are analogous to a handover task, whereby: 1) the end-effector must find a pose that is appropriate for grasping the object without colliding with the hand, 2) moving to the object and closing its gripper to grasp the object, and 3) retracting whilst keeping its gripper closed to complete the handover task. Example keyframes are illustrated in Figure 3.1.

However, utilizing this heuristic using the demonstrations collected from the HandoverSim produces disproportionate keyframes than what would have been desired to capture bottleneck actions. Figure 3.2 provides an example where this heuristic fails to identify appropriate keyframes. The reason for this is that the closing motion of the gripper is very slow within the simulation, which flags the gripper state as not changed and joint-velocities near zero. Furthermore, the transition from approaching to grasping is not captured, despite being a crucial bottleneck for determining a collision-free pre-grasp pose.

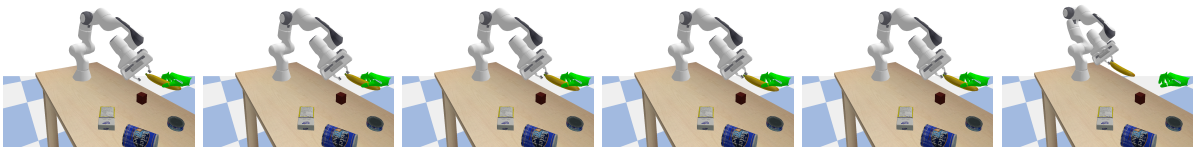
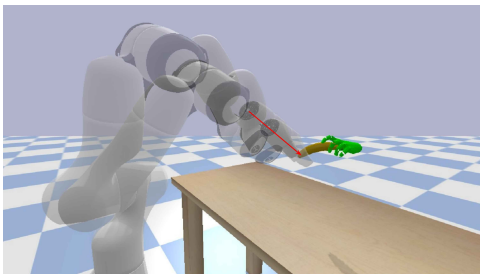
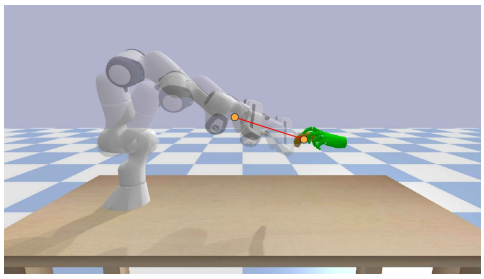


Figure 3.2: Failed generation of handover keyframes using a heuristic: 1) the joint-velocities are near zero, 2) the gripper open state has not changed

The extraction of good keyframe actions is essential to yield correct learning of the (handover) task [31]. Therefore, to prevent the redundant generation of keyframes from the heuristic, 1) the approach object phase of Yang et al. [27] was utilized to identify the first keyframe, and 2) the grasping triggered by the policy was utilized as the second keyframe. Using the phases, the end-effector is only moving forward from the approach phase to safely grasp the object as illustrated in Figure 3.3a. Finally, a hand-coded policy is used to move from grasp to the goal, similar to the implementation of Yang et al. [27] in HandoverSim. Hence, the handover task must capture the policy's ability to safely acquire the object from the giver's hand (no collision) and predict a stable grasp when moving to the goal location (no drop).



(a) Angled perspective of the handover.



(b) Side angle perspective of the handover.

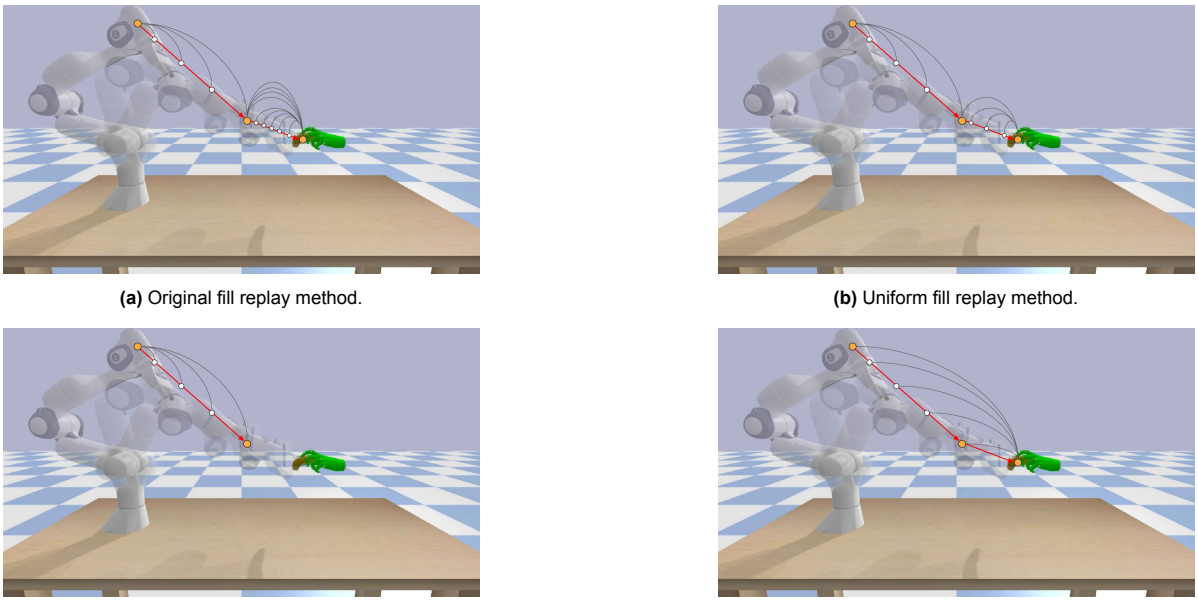
Figure 3.3: Transition phase from pre-grasp/approach to grasping the object. The red line indicates a forward (from the end-effector's perspective) movement for this transition phase.

3.1.2. Sampling methods

Concerning the extracted keyframes, various sampling methods for the replay buffer are analyzed to identify data imbalance for input-action pairs. Each alternative sampling method was found by iteratively analyzing the sampling method. The sampling method is explained below and illustrated in Figure 3.4.

1. **Original fill replay**: Samples are stored in the replay buffer in conjunction with subsequent keyframes. Adding several copies of the subsequent keyframes indicates important phase transitions and makes them more likely to be sampled.

2. **Uniform fill replay**: Samples are stored at a fixed rate between each transition phase and do not re-add samples between keyframes. This avoids emphasizing transition phases.
3. **Only learn approach fill replay**: This fill replay method contains input-action pairs with actions till the approach keyframe.
4. **Cropped fill replay**: Contrary to the only learn approach fill replay, input-action pairs are sampled until the approach phase, but all actions point towards the grasping action.



(a) Original fill replay method.

(b) Uniform fill replay method.

(c) Only learn approach fill replay method.

(d) Cropped replay method.

Figure 3.4: Sampling methods for filling the replay buffer. Figure 3.4a is the original sampling method used by ARM [33]. Figure 3.4b adapts the original sampling method by avoiding oversampling transition phases. Figure 3.4c samples up to the approach phase of the handover task with the keyframe action at the approach. Figure 3.4d also uses samples up to the approach phase but its actions at the grasp location.

To evaluate PerAct for its training stage (the top/gray box in Figure 2.2), the output is compared using Equation 3.1 and Equation 3.2. These equations calculate the Euclidean and cosine distance between translation and orientation. This allows for evaluating the performance of PerAct to some reference action pose and shows whether those sampling methods have yielded correctly learned behaviors.

$$\epsilon(x, \hat{x}) = \|x - \hat{x}\|_2 \tag{3.1}$$

$$\epsilon(q, \hat{q}) = \arccos\left(|\langle q, \hat{q}\rangle|\right) \tag{3.2}$$

3.2. Multi-camera setup

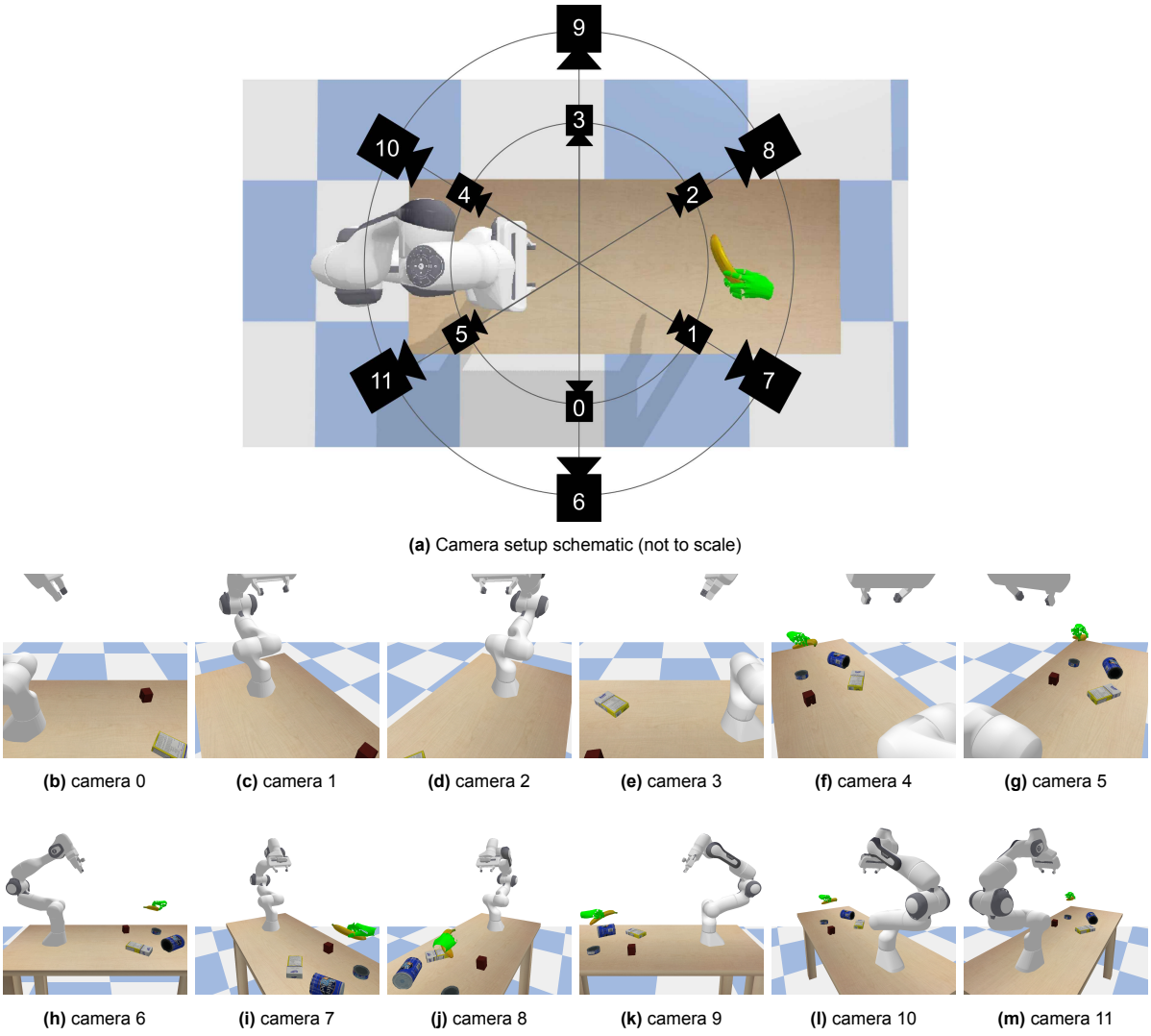
The multi-camera setup contains 12 cameras covering the goal region where the HandoverSim completes its handover task. This setup contains 6 near and far cameras at horizontal radii of 0.5 meters and 1.2 meters, respectively, and 0.3 meters higher than the goal region. The cameras have a resolution of 128x128, akin to PerAct. Using a combination of the 12 cameras, 5 configurations were analyzed for the impact on the generated voxels.

- **Setup 1 - cameras 6, 8 10** uses 3 far cameras. The hypothesis is that this captures a good representation of the task scenery, but fails to capture the details of the handover objects, leading to unfilled voxels of the object. This is suspected to yield lower grasping precision.
- **Setup 2 - cameras 6, 7, 8, 9, 10, 11** uses 6 far cameras. The hypothesis is similar to setup 1. but covers the handover object in more detail. A higher grasping precision is expected.
- **Setup 3 - cameras 0, 1, 2, 3, 4, 5** uses 6 close cameras. The hypothesis is the closer cameras capture the object in more detail but fail to understand the task scenery.
- **Setup 4 - cameras 1, 3, 5, 6, 8, 10**: uses 6 cameras, alternated by close and far. The hypothesis

is that this balances setting 2 and 3, capturing the essence of the handover and having a detailed description of the object.

- **Setup 5 - cameras 4, 5, 6, 7, 8, 9**: uses 2 cameras that are near, close to the robot, and far at the object space: Observing the HandoverSim shows that the handovers take place on the right of the goal region. It is hypothesized that this camera setting captures the object and scenery best at the location where the handover takes place.

The setup is visualized in Figure 3.5, as well as examples of the image observations from the cameras. It is hypothesized that providing a higher quality of the handover grasp scene should yield better contextuality of the handover scene. The expected result is better learning of the handover task. Especially for the case of finer grasping locations, obtaining a detailed description of the handover object is expected to have increased grasping accuracy.



(a) Camera setup schematic (not to scale)

(b) camera 0 **(c)** camera 1 **(d)** camera 2 **(e)** camera 3 **(f)** camera 4 **(g)** camera 5

(h) camera 6 **(i)** camera 7 **(j)** camera 8 **(k)** camera 9 **(l)** camera 10 **(m)** camera 11

Figure 3.5: Camera setup for collecting expert demonstrations. Figure 3.5a gives an overview of the camera placements with corresponding camera IDs. The cameras cover the handover goal location with a pre-specified radius and are positioned 0.3 meters above the goal location. Figures 3.5b-g observe the handover space from a nearby (0.5 meter) perspective to capture detail, while Figures 3.5h-m observe the handover space from far (1.2 meters) perspective to capture the scene. The camera locations in the figure are not to scale

To evaluate the grasping accuracy, the trained PerAct model is evaluated in the HandoverSim using the various camera setups. Aside from the high-level task completion defined in HandoverSim [25], the root mean-square error (RMSE) is introduced to capture grasping accuracy with some reference

grasping set. Equation 3.3 takes the average of each grasp prediction, similar to Equation 3.1, but this metric is used cross sequences. The RMSE is only calculated when the end-effector reaches the predicted grasp pose.

$$\text{RMSE}(x, \hat{x}) = \sqrt{\frac{\sum_{i=0}^{N-1}(x_i - \hat{x}_i)^2}{N}} \tag{3.3}$$

3.3. Handover objects

The S1 split in HandoverSim captures 35 and 5 demonstrations for training and validation, respectively. Table 3.1 provides all successful attempts per YCB object for the train and test split.

Table 3.1: Available demos per YCB object, collected using Yang et al. [27].

Object	Train	Validation
Banana	31	4
Bleach cleanser	22	3
Bowl	22	4
Cracker box	24	4
Extra large clamp	20	3
Foam brick	26	4
Gelatin box	26	4
Large marker	23	4
Mug	22	1

Object	Train	Validation
Mustard bottle	16	3
Pitcher base	18	4
Potted meat can	22	3
Power drill	22	3
Pudding box	31	2
Scissors	21	3
Sugar box	24	4
Tomato soup can	16	3
Tuna fish can	17	1

The handover per YCB object is evaluated in the HandoverSim. The failure modes are further distinguished by {*with grasp*, *without grasp*}. This separation is made by whether a grasping attempt is made or not, and provides insights into what stage the failure is met. In particular, italicized failure types are obtained with valid grasps. Moreover, Equation 3.3 is also calculated for the evaluation of each object to discover the relation between the object properties and the grasping accuracy.

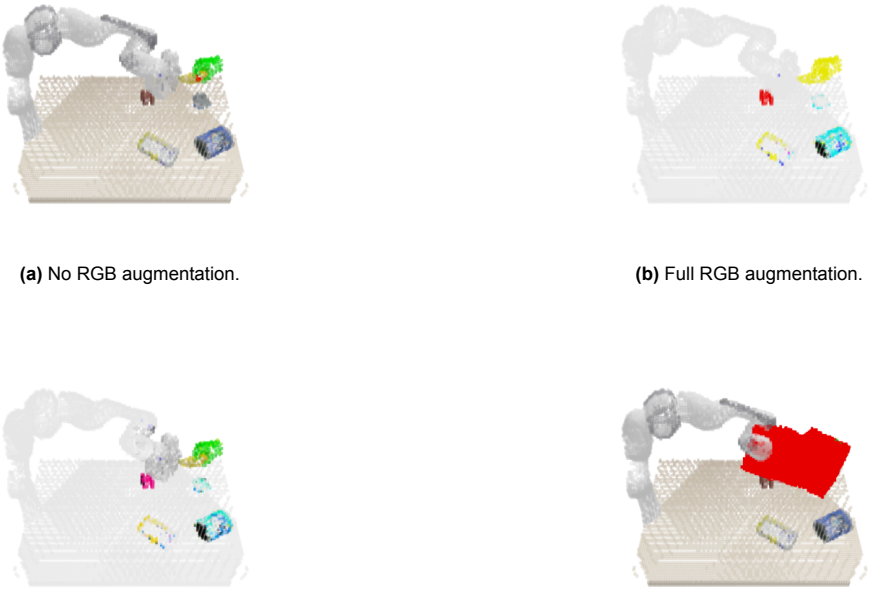
Table 3.2: Description of failure types at grasp and non-grasp stages. The italic failure types indicate all failures when valid grasp predictions are obtained.

Stage	Failure Type	Description
Without Grasp	*Contact*	Whilst moving to the predicted grasp pose, the gripper is in contact with the hand.
	Drop	Whilst moving to the predicted grasp pose, the gripper finger pushes the object out of the hand.
	Timeout	Infeasible prediction.
With Grasp	*Contact*	When closing the gripper, the gripper is in contact with the hand.
	Drop	Gripper was able to contact the object, but could not establish a stable grasp.
	Timeout	The predicted grasp fails to grasp the object.

3.4. Voxel-based RGB augmentations

The voxel RGB augmentations are aimed at creating a model that is robust against distractors. The RGB augmentation implementations are distinguished by 1) full RGB augmentation and 2) partial RGB augmentation. The full RGB augmentation relates to the augmentation of the full voxel observation scene, while the partial RGB augmentation relates to the augmentation of the full scene except for the handover object and the gripper. Figure 3.6 illustrates each RGB augmentation in the voxel observation.

For each RGB camera image, augmentation is applied using Pytorch's colorjitter function [40]. This function adjusts the RGB values randomly on brightness, contrast, saturation, and hue scales. For each sample used in training, the scaling factor has a range $\pm$ 0.2, 0.2, 0.2, and 0.1, respectively. The full RGB augmentation applies this method for all cameras that are converted to voxel observation. On the other hand, the partial RGB augmentation avoids voxels surrounding the end-effector and the handover object. The locations for these are obtained from the HandoverSim. A space of 20 indices along each direction, oriented with the object's coordinate system, is selected whose voxel features remain unchanged. During the training process, the augmentations can be turned on, while turned off during the validation at iterations to keep the observations similar to HandoverSim.



(a) No RGB augmentation.

(b) Full RGB augmentation.

(c) Partial RGB augmentation.

(d) Partial RGB augmentation bounds.

Figure 3.6: RGB augmentation strategies applied to voxel observations. Figure 3.6d displays the borders at which no RGB augmentations are applied in red (which cover the gripper and the handover object). The scaling factor for the augmentations in this example is amplified to enhance the result.

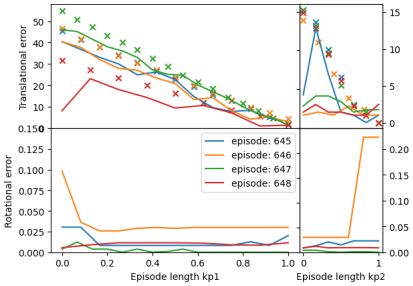
The performance of voxel-based RGB augmentation is evaluated on the HandoverSim Benchmark, using a similar evaluation method as described in section 3.3. This analyzes the success rate and grasping accuracy with the object properties.

4. Results

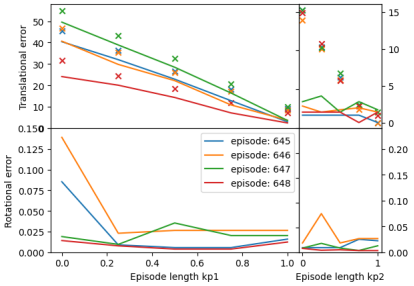
4.1. Expert demonstration: keyframe extraction and sampling

The validation set for the banana handover was used to compare each sampling strategy, and the results are provided in Figure 4.1. Each lineplot (Figure 4.1a,b,d,e) shows the translation error at the top and rotational error at the bottom using Equation 3.1 and Equation 3.2, respectively. The translational error is described in terms of voxels. The translational error comparing the expert action with the prediction is plotted using a solid line in Figure 4.1a, 4.1b, 4.1d, and a dashed line in Figure 4.1d. The solid line in Figure 4.1d compares with a ground-truth reference set from Eppner et al. [28] [1]. The datapoints in the plots are ordered chronologically to identify at what stage PerAct fails to predict appropriate actions for the handover task. Hence, a handover episode starts from the left (where the end-effector starts moving, illustrated in Figure 2.1d) and ends on the right (where the end-effector grasps the object, illustrated in Figure 2.1e).

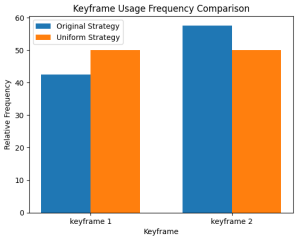
.



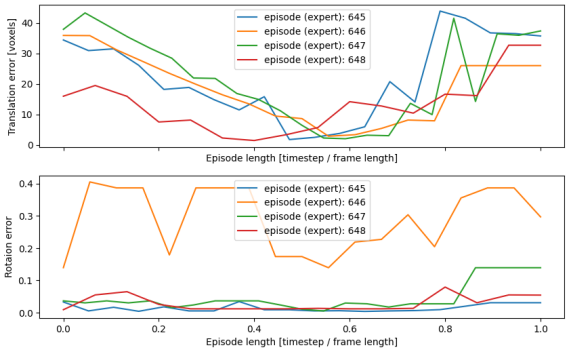
(a) Original fill replay method



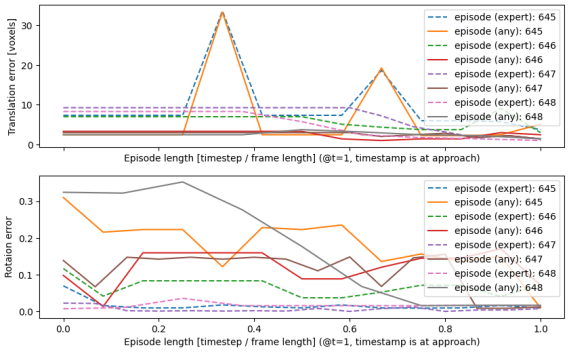
(b) Uniform fill replay method



(c) Proportional keyframe sampling frequency for original (blue) vs. uniform (orange) method



(d) Only learn approach fill replay method



(e) Cropped fill replay method.

Figure 4.1: Error metrics for different sampling methods for filling the replay buffer, which have been described in section 3.1. The top graph shows the translational error in discretized (i.e., voxel) distance, and the bottom graph shows the rotational error. Figure 4.1e shows a dashed and solid line. The dashed line displays the error for the expert action. The straight line shows the true error for all available grasps. The × in Figure 4.1a and 4.1b is the distance error for the gripper pose at that sample.

[1] Throughout the paper, ground-truth refers to the robust grasps generated from Eppner et al. [28]

Specific to the **original and uniform fill replay methods**, the plots in Figure 4.1a and 4.1b are divided into the approach phase (kp1) and grasp phase (kp2) to identify prediction errors specific to these phases. Hence, the left-hand side has the approach actions, and the right-hand side has the grasp action as a reference for calculating the errors. The vertical line in the middle indicates the transition point. Additionally, the gripper distance from the keyframe action at this sample point is given by $\times$ in both figures.
The left-hand side of Figure 4.1a and 4.1b yields a decreasing translational error at the approach (kp1) phase. This indicates the model can slowly converge toward the key action as the gripper reaches the approach. While this method is stable, the desired action should have a translational error at zero. The right-hand side of Figure 4.1a and 4.1b shows that the gripper can reach the grasp pose more consistently with the uniform method.
Analyzing the keyframe action frequency of both methods, Figure 4.1c compares the proportional keyframe sampling frequency per approach and grasp action. This confirms that the grasp keyframe is more emphasized in the original fill replay method (blue) than the approach keyframe, whereas the uniform fill replay method (orange) has even weights on both keyframes.
Furthermore, comparing the $\times$ with the prediction shows that both fill replay methods have similar distances from the keyframe action.

The only learn approach fill replay method is used to predict the approach directly, rather than learning the transition phases of a handover (approach and grasp). Figure 4.1d examines the error considering the approach action and shows an increasing error as the observation reaches closer to the approach keyframe. Similar to the original and uniform fill replay method, this method does not yield a low error across its demonstrations.

The cropped fill replay method learns to predict the object grasp by using samples till the approach phase, instead of learning the approach. Contrary to previous approaches, Figure 4.1e demonstrates a consistently low error relative to the expert action (dashed line). Furthermore, the prediction can get much closer to one of the ground truth grasps (solid line).

4.2. Multi-camera setup

The multi-camera setup relates to the coarseness (or fineness) of the voxel observation, which is used for learning the expert actions and for inferring new observations. The voxel observation resolution is calculated by the number of occupied voxels in the $100\times100\times100$ voxel space. By dividing the occupied voxels over the full voxel space, a ratio is calculated that relates to how much information from the scenery is captured by each camera setup introduced in section 3.2.
The table in the full scene is present in the voxel observation, which could yield a high value of occupied voxels. For this reason, the occupied voxels for the handover object are considered to relate to the information captured for the task object. The occupied voxel space for this instance is calculated by using the ground truth object location and spanning a box region of $20\times20\times20$ voxels around the object (in a similar method as how the RGB augmentation was applied, but without applying rotation).

Table 4.1: Voxel resolution for each camera setup using occupancy. Each voxel is assigned by an RGB value (voxel feature) or kept empty. The camera relation shows that for setup 5, the density is highest, especially for the handover object.

		Full scene		Object (banana)	
	Used Cameras	Occupied voxels	Percentage occupied voxels	Occupied voxels	Percentage occupied voxels
1	6,8,10	9277	0.9 %	178	3.2 %
2	6,7,8,9,10,11	14026	1.4 %	243	4.3 %
3	0,1,2,3,4,5	10750	1.1 %	75	1.3 %
4	1,3,5,6,8,10	14738	1.5 %	200	3.6 %
5	4,5,6,7,8,9	15565	1.6 %	263	4.7 %

To identify the relation between voxel observation resolution and camera setup, episode 46 of the `banana` handover was used as an example, and its voxel occupancy is provided in Table 4.1. This

episode is one of the edge cases where the handover is located near the scene boundary and requires a camera placement that can still capture (object) task information. The results are calculated by averaging the occupied voxels over all samples (from the replay buffer) of the episode. Table 4.1 shows the highest occupied voxels for camera setup 5, with 6 cameras used in near-far configuration, mainly located to the right to capture the objects. Appendix A provides the voxel observation for each setup and shows that for camera setup 2, 4, and 5, the banana is represented more clearly in the voxels. The banana for camera setup 3 is almost not visible in this example.

Furthermore, PerAct was trained with the various camera setups and inferred in HandoverSim using the same camera setup. The results are provided in Table 4.2 and show that, despite the higher resolution for camera setting 5, the overall success rates for train, val, and test are highest for camera setting 2. The positional RMSE (to the ground truth) for the test set is calculated using Equation 3.3 to obtain the accuracy of the trained model using the camera setup. However, this indicates that between every camera setup, the accuracy variance is very little and has an accuracy of roughly 0.03 m.

Table 4.2: Success and failure rate for every camera setting, evaluated for the training/validation/test split for s1 banana. The highest success rates are highlighted in bold. The last column provides the root mean-square error (RMSE) to compare the location of the predicted grasp with the ground-truth reference set.

	train				val				test				
camera setting	success %	failure %			success %	failure %			success %	failure %			RMSE: Δ_{GT} (m)
		contact	drop	timeout		contact	drop	timeout		contact	drop	timeout	
1	28.57	2.86	37.14	31.43	20.00	0.00	80.00	0.00	**20.00**	0.00	50.00	30.00	0.025
2	**54.29**	5.71	28.57	11.43	**40.00**	0.00	60.00	0.00	**20.00**	10	20.00	50.00	0.03
3	31.43	5.71	34.29	28.57	**40.00**	0.00	60.00	0.00	0.00	30.00	30.00	40.00	0.028
4	48.57	2.86	31.43	17.14	0.00	20.00	0.00	80.00	**20.00**	10.00	20.00	50.00	0.026
5	31.43	8.57	34.29	25.71	**40.00**	0.00	40.00	20.00	10.00	10.00	0.00	80.00	0.03

4.3. Handover objects

4.3.1. Grasped objects properties

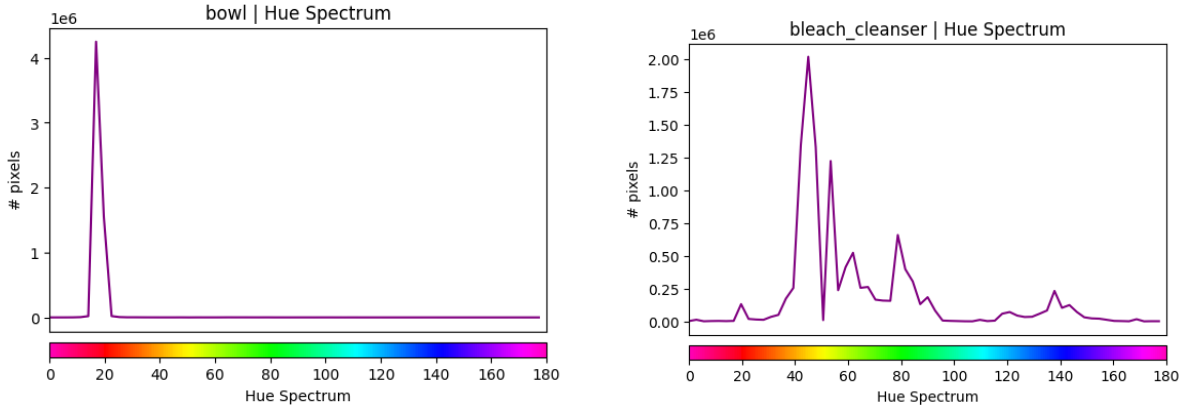
Table 4.3 provides object properties by object dimensions, volume, texture complexity, and shape complexity [2]. Object dimension is obtained from YCB [37]. For this property, the *critical dimension* is highlighted in bold for each object, which represents the smallest width (disregarding its geometrical shape) for the object to form a grasp. The *critical dimensions* for the bowl, mug, and pitcher base are not highlighted in Table 4.3. The grasp opportunities for these items are located at the rim and/or the handle, which is significantly smaller than the object's dimensions. The *volume* of the object is approximated by a bounding volume ($\cong$ width $\times$ length $\times$ height). *Texture complexity* is estimated through the mono- or multimodality of the color spectrum of the object's texture, which yields a simple or complex texture, respectively. An example hue spectrum is illustrated in Figure 4.2 with single vs. multiple peaks. This spectrum is obtained by converting the RGB texture mapping onto a hue, saturation, and brightness value. Here, an assumption is made by ignoring the saturation and brightness, which removes white and black textures. Lastly, *shape complexity* is calculated by measuring the entropy of the curvature of the object's 3D mesh [41]. Sukumar et al. [42] argue that this metric quantifies (human) perceptual complexity of objects and encapsulates it as an important feature for object recognition and retrieval.

[2] Table 4.3 also contains the column *Coverage*, which is not an object property. This is a data distributional property and is introduced in chapter 5

Table 4.3: YCB object properties: dimension, volume, texture complexity, shape complexity, and data similarity (Coverage) between the training and test set.

Object	Dimensions (mm) [a]	Volume (L) [b]	Texture complexity	Shape complexity	Coverage(train, test) [d] 15 deg - 20 deg
Banana	**36** x 190	0.25	simple	4.43	0.1 - 0.3
Bleach cleanser	250 x 98 x **65**	1.59	complex	4.39	0.2 - 0.2
Bowl	159 x 53	1.34	simple	**6.17**	**0.4 - 0.5**
Cracker box	**60** x 158 x 210	1.99	complex	3.18	0 - 0
Extra large clamp	165 x 213 x **37**	1.30	complex	5.95	0.1 - 0.2
Foam brick	**50** x 75 x **50**	0.19	simple	3.78	0.2 - 0.2
Gelatin box	**28** x 85 x 73	0.17	complex	3.36	0.3 - 0.3
Large marker	**18** x 121	0.04	complex	4.33	0 - 0
Mug	80 x 82	0.52	simple	5.81	0.3 - 0.4
Mustard bottle	**58** x 95 x 190	1.05	simple	4.42	0.1 - 0.2
Pitcher base	108 x 235	2.74	simple	5.17	0.1 - 0.1
Potted meat can	**50** x 97 x 82	0.40	complex	*5.17*	0 - 0.1
Power drill	**35** x 46 x 184	0.30	complex	4.54	0 - 0
Pudding box	**35** x 110 x 89	0.34	complex	3.33	0 - 0.1
Scissors	87 x 200 x **14**	0.24	complex	5.14	0.1 - 0.1
Sugar box	**38** x 89 x 175	0.59	complex	3.17	0 - 0.1
Tomato soup can	**66** x 101	0.44	complex	*5.8*	0 - 0
Tuna fish can	85 x **33**	0.24	complex	*5.09*	0.2 - 0.4

[a] The smallest dimension (disregarding its geometrical shape) for each object is highlighted in bold. Objects with a specific grasping opportunity smaller than the object's dimension are not highlighted.
[d] Coverage is not an object property, but is a data distributional attribute between train and test data.



(a) Hue spectrum of bowl.

(b) Hue spectrum of bleach cleanser.

Figure 4.2: Object texture complexity, indicated by modality. Simple textures contain a single peak (such as Figure 4.2a), while complex textures contain multiple peaks (such as Figure 4.2b).

Table 4.4 provides the success and failure rates from HandoverSim for each object in the test set.

4.3.2. Grasped objects

Table 4.4 provides an overview of all objects evaluated in the HandoverSim. The bowl has the highest success rate among all objects. Failure due to *contact* is the highest for the potted meat can and is evenly attributed to both during grasping and before grasping. For the pitcher base, all sequences have led to a *drop*, of which 70 % is with a grasping attempt. *Timeout* is highest for the scissors, which are post-grasping. In general, failure due to contact is not the highest occurring failure per object, and timeout without grasping is low for all objects. Comparing the RMSE for grasp poses used at training (Δ_{train}) and ground truth (Δ_{GT}), the predictions yield closer values with the ground truth.

Additionally, failure rates for each YCB object (ordered by *critical dimension*) are plotted in Figure 4.3 with a trendline for *drop* and *timeout*. The plot only includes failures when valid grasp predictions are made, which omits timeout *without grasp*. Objects with specific grasping opportunities are omitted. The

trendlines indicate that objects with increasing *critical dimension* yield higher failures due to a drop, but decreased failures due to timeouts. Figure 4.3 illustrates two examples of timeout (left) and drop (right). The gripper is in contact with the object with the higher *critical dimension*; the gripper misses contact with the object with the lower *critical dimension*.

The relation of grasp opportunity and RMSE for each object is illustrated in Figure 4.4a. The graph shows that the highest RMSE is obtained from the pitcher base, which has its grasping opportunity at the handle. The mug has its grasping opportunity at the handle and rim, but yields a low RMSE. Conversely, the *critical dimension* for the tomato soup can is the highest, but its RMSE is among the lower values. Furthermore, inspecting the RMSE with object volume in Figure 4.4b shows that the predictions can reach much closer to the grasp poses for smaller objects.

Table 4.4: Success and failure rate for every object. Evaluated for the training/validation/test split for s0. The bowl has the highest success rate. The highest success/failure/RMSE are indicated in bold.

	summary				with grasp			without grasp			$RMSE_{grasp}$	
Object	success %	failure %			failure %			failure %			Δ_{train} (m)	Δ_{GT} (m)
		contact	drop	timeout	contact	drop	timeout	contact	drop	timeout		
banana	30	0	30	40	0	10	40	0	20	0	0.064	0.024
bleach cleanser	10	30	40	20	0	30	20	30	10	0	0.111	0.095
bowl	**50**	0	30	20	0	0	20	0	30	0	0.079	0.047
cracker box	20	0	80	0	0	30	0	0	50	0	0.1	0.091
extra large clamp	10	0	60	30	0	0	30	0	60	0	0.139	0.122
foam brick	20	10	50	20	0	50	20	10	0	0	0.062	0.024
gelatin box	20	20	10	50	0	10	50	20	0	0	0.078	0.04
large marker	30	30	10	30	20	10	30	10	0	0	0.052	0.025
mug	20	20	30	30	10	10	20	10	20	10	0.074	0.03
mustard bottle	0	30	60	10	0	0	0	30	60	10	0.061	0.043
pitcher base	0	0	**100**	0	0	70	0	0	30	0	**0.19**	**0.157**
potted meat can	0	**40**	60	0	20	40	0	20	20	0	0.068	0.048
power drill	30	20	30	20	0	10	10	20	20	10	0.065	0.04
pudding box	20	0	40	40	0	10	20	0	30	20	0.047	0.024
scissors	10	10	20	**60**	0	10	60	10	10	0	0.074	0.034
sugar box	10	0	60	30	0	10	30	0	50	0	0.098	0.09
tomato soup can	20	10	60	10	0	40	10	10	20	0	0.067	0.036
tuna fish can	20	20	20	40	10	20	30	10	0	10	0.047	0.02

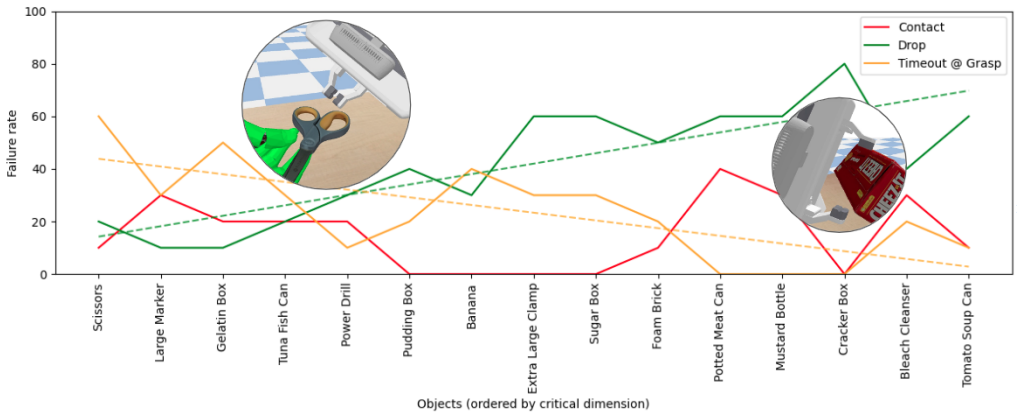
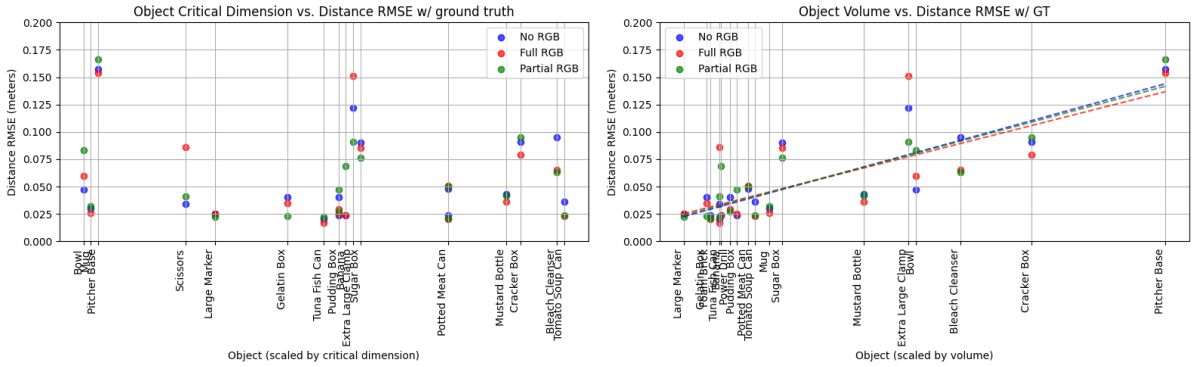


Figure 4.3: Failures for each YCB object ordered by *critical dimension* (excluding objects with specific grasp opportunities). Overlayed with timeout for scissors and drop for cracker box. Timeout failure *without grasp* is excluded to identify failures with valid predictions.



(a) RMSE with ground truth grasps vs. YCB object (grasp opportunity).

(b) RMSE with ground truth grasps vs YCB object (size).

Figure 4.4: Effects of voxel none vs. full vs. partial RGB augmentation. Plotting the RMSE error vs. the *critical dimension* and ground truth grasps for each YCB object, evaluated on S1 test split.

Figure 4.5 shows the training translational losses per object for simple and complex textures. The graph for objects with simple textures shows a gap among these objects, whereby some objects obtain a lower loss at the 6000th iteration (~4 vs. ~5). This gap is not visible for objects with complex textures, where the losses are spread between a 3-6 range. Moreover, the trendline at which losses decay is similar for both simple and complex textures.
The bowl has the highest shape complexity and obtains the highest success in the handover benchmark. Despite the extra large clamp having the second-highest shape complexity, the success rate for this object is only 10 %. Furthermore, the impact of shape complexity (comparing the bottom and top 3 shape complexities) shows varying results on the training translational loss in Figure 4.5.

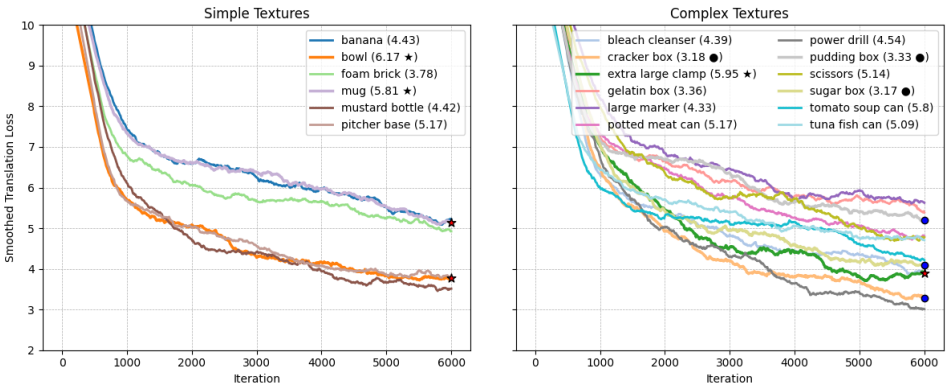


Figure 4.5: YCB object training loss vs. texture complexity and shape complexity. Simple and complex textures are separated by the left and right plots, respectively. The top 3 (★) and bottom 3 (●) shape complexities are highlighted on the legend and plot.

4.4. Voxel-based RGB augmentation

The success rates of the handover benchmark for each voxel RGB augmentation are given in Figure 4.6[3], with varying results for applying RGB augmentations. The full augmentation yields the highest increase for the tuna fish can, pudding box, and pitcher base. The partial augmentation yields the highest increase for the sugar box and bowl. For some objects, either augmentation yields worse performance.
The RMSE obtained for each voxel RGB augmentation is plotted in Figure 4.4 against *critical dimension* and object volume. The trendline for each RGB augmentation shows a similar trend where an increased volume results in an increased RMSE. While the trend line for volume vs. RMSE remains the same, the improvements in RMSE do not correlate with the improvements in success rates in the benchmark. For instance, the partial RGB augmentation for the extra large clamp yields a decreased

[3]The horizontal line for 20 ° coverage ratio is introduced in chapter 5. The coverage ratio for each object is found in Table 4.3

RMSE, but this improvement is not seen in the success rate. Similarly, the RMSE for the pitcher base is approximately the same, but the success rate is improved for both augmentations.

The complete success and failure rate for every object per voxel-based RGB augmentation has been added in Appendix B.

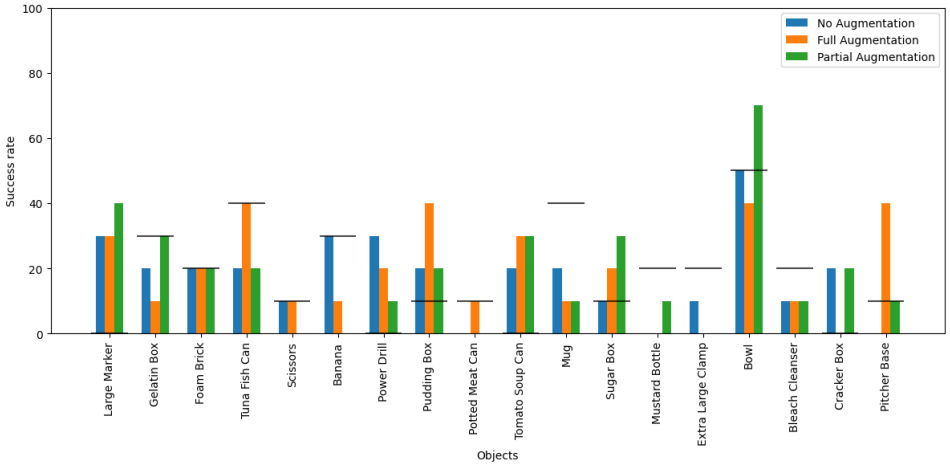


Figure 4.6: Success rates of YCB objects with (no/full/partial) RGB augmentations, ordered by volume. The horizontal line is obtained from the 20° coverage ratio in Table 4.3

5. Discussion

The overarching research question is whether *a BC agent can learn a human-to-robot object handover using an end-to-end framework by learning from demonstrations*. To answer this question, the discussions look into the methods for using the behavior cloning agent for the handover task: how expert demonstrations should be provided to the agent through input-action samples and the experimental setup; experiments using various handover objects in the handoversim and voxel-based RGB augmentations. Finally, the advantages and shortcomings of the end-to-end method are discussed by comparing methods requiring a perception component.

Expert demonstrations

Firstly, the approach and grasping are selected as bottleneck keyframe actions in which the end-effector must transition to perform the handover task. Specifically, the approach phase is required as an intermediate trajectory point for the motion planner to generate a collision-free path to the final grasp.
Using these keyframes, the optimal sampling methods were looked into for generating input-action pairs. The *original and uniform fill replay* methods fail to learn an action towards the approach phase as found in Figure 4.1a and 4.1b. Rather, the predicted actions are near the end-effector of the sample and provide a forward movement when observing in the HandoverSim. It was hypothesized that due to the copies of subsequent keyframes (here: approach to grasp), PerAct overemphasized learning a forward movement behavior. From the (voxel) observation, the action from approach to grasp is near the end-effector, which likely results in the policy learning attention from the gripper, rather than the relation between the gripper and the object. The *uniform replay buffer* avoids overemphasizing the latter transition but yields similar forward movement behavior.
Similar results have been observed in the works of Guo et al. [43] and have been attributed to transition with non-optimal actions. Hence, the inclusion of the second transition phase likely produces a local minimum, where the policy gets stuck at a certain sub-optimal policy to learn a forward movement.
Using this result, the *only learn approach* looks into learning to predict an action directly to the approach. However, the policy fails to learn the grasp-object relation due to the large distance gap between the action and the object.
Ultimately, the *cropped replay* method works best by avoiding samples that produce a local minimum and learning the grasp-object relation.

Multi-camera setup

Secondly, the camera setup that yielded the highest success was obtained from all cameras positioned far. According to Kasaei et al. [44], the optimal camera pose offers the greatest scene coverage to obtain an accurate view of the object. Considering the occupied voxels per camera setup in Table 4.1, the optimal camera setup should be obtained from camera setup 5. However, inspecting the observations obtained from the near cameras for camera setup 5 shows that these cameras can be occluded by the robot arm itself. When fusing these RGB-D camera images to voxel observations, these could contain rough representations of the handover object similar to Figure A.1e from camera setup 3.
This analysis is in line with the works of James et al. [45], where cameras are positioned to provide maximum coverage over the task space while minimizing occlusion due to the robot arm. Furthermore, a wrist camera is used in combination with cameras surrounding the task scene.
Conversely, Dai et al. [46] propose to use a camera setup by moving the wrist camera itself to prevent occlusion due to the robot arm, whilst benefiting from great scene coverage. However, for a handover

task, the generation of this setup can be costly. Instead, Christen et al. [30] provide a framework that works using the wrist camera only to provide an egocentric perspective. A promising alternative is to use a similar method as from the works of Dai et al. [46] by solely using a wrist camera but fusing camera images during the approach phase to capture an accurate representation of the object.

Experimenting with handover objects and voxel-based RGB augmentations

Thirdly, experiments using handover objects are analyzed to identify object properties that could yield better success for a handover. Specifically, the object's grasping opportunity width (*critical dimension*), volume, texture complexity, and shape complexity are analyzed.

Inspecting the RMSE with object properties showed a positive relation for volume with the grasping error. Additionally, an increasing *critical dimension* yields increased object drop failure and decreased timeout failure. This can be attributed to converting discretized actions to the original coordinate frame, which produces a resolution error yielding lower precision [31]. As mentioned by Mousavian et al., a small perturbation in the grasp pose can transform a successful grasp into a failure [47]. Specifically, smaller objects and objects with specific grasp opportunities (such as the pitcher base) require high-precision for grasping the object to avoid missed grasps (timeout failure post grasping). Additionally, objects with high *critical dimension* require high-precision to not contact the gripper fingers with the object to avoid object drop.

Besides analyzing object properties, the data distribution for objects in training and testing was inspected through coverage and is provided in the last column of Table 4.3. (Grasp) coverage captures the distribution of sampled grasps with some reference ground truth [28]. Using this metric with object-orientation provides insight into the train-test data distribution using Equation 5.1. Specifically, an object orientation from the test set, $q_{test} \subset O_{test}$, is covered if there exists some orientation $q_{train} \subset O_{train}$ that is within angled distance σ_q. Chao et al. [34] uses σ_q = 15°, but considering the voxel observations are augmented by 45° during training, a coverage criteria of σ_q = 20° is also provided. Notably, the coverage only uses object orientation and does not consider shape symmetry or position, which could shift the coverage value.

$$\arccos(|q_{test}, q_{train}|) \leq \sigma_q \tag{5.1}$$

Figure 4.6 displays the coverage with success rate. The success rate yields comparable results with the coverage for most objects per RGB augmentation. The collected expert demonstration for training is very sparse with limited object-orientations, leading to narrowly learned grasping behavior. This observation aligns with the works of Belkhale et al. [48], which states that imitation learning agents struggle with distribution shifts due to limitations in the quality of expert demonstrations. Similarly, behavior cloning suffers from 'compounding error', where a small deviation from the expert demonstrations could yield completely different behaviors [49]. However, considering low failure due to timeout without grasp in Table 4.4, predicted actions can reach the object, showing that it does not suffer from 'compounding error'. To improve handovers for the behavior cloning agent, future work could rely on works from GenH2R [50] and SynH2R [51] to expand the convex hull of training data.

Furthermore, objects can be grasped from multiple orientations. Using Equation 2.1 to supervise a single expert grasp orientation, the model undermines non-covered grasps. Therefore, the method tends to overfit to those grasp scenarios, limiting generalization. As referenced by Shridhar et al. [31], Equation 2.1 could be adapted to an energy-based method [52] to allow for a more generalizable multi-modal model.

Comparing end-to-end vs. perception control framework

Finally, a comparison is made with the end-to-end method used in this research and methods evaluated in HandoverSim [19] [27] [30], which require a perception component. Specifically, the end-to-end framework can learn and predict actions from observation directly, as opposed to methods with separate perception-planning components that require training the hand-object segmentation model [27], besides the grasping network.

Chisten et al. [30] identify that the perception component suffers from 1) camera noise, 2) body tracking and hand segmentation errors, 3) viewpoint change (wrist to external), and 4) unseen human behaviors. Rather, the end-to-end method introduced here avoids the perception component using RGB-D observations only to generate a task scene representation, which avoids limitations by 2 & 3, and prevents error propagation [1]. Moreover, Table 4.4 shows that the failure due to contact is not the dominant

failure, signifying that PerAct learns to distinguish the object from the hand. This indicates that the end-to-end framework can discover features for the handover task by avoiding contact with the hand as mentioned by Levine et al. [1] without requiring a perception pipeline. Despite avoiding error propagation from the perception pipeline, the PerAct model suffers from inaccuracies introduced by the voxelization that can lead to failures in HandoverSim.

6. Conclusion

Current visuomotor manipulators use an end-to-end framework to train the perception-planning-action components simultaneously to avoid error propagation. Within this research field, grasping has shown capabilities to deal with increasingly complex environments, owing to benchmarks. Although human-in-the-loop is limited due to reproducibility and costly sampling time, object handovers have shown progression by adapting grasping methods to work alongside hands grasping the object. These methods utilize an end-to-end grasping framework but require a perception pipeline for hand-object segmentation. Given the challenges and limitations of prior handover methods, this paper assesses an end-to-end behavior cloning agent for the handover task and does not require a perception component. Moreover, a HandoverSim simulation is created to benchmark with standardized and reproducible receiver policies. Specifically, this paper addresses: **assessing an end-to-end behavior cloning agent for discretized 6-DoF manipulation via RGB voxel inputs in a human-to-robot object handover benchmark**.

The method uses a Perceiver-Actor agent that allows learning from expert demonstrations. The method works by fusing RGB-D camera observations to voxels and works within this frame to output discretized actions. From the benchmark, a reference policy was used for collecting expert demonstrations and allowed for experimenting with the optimal data acquisition strategy. The experiments analyzed selecting keyframe actions and sampling methods, camera setups, inspecting handover object properties, and applying voxel-based RGB augmentations.

The sampling methods showed that input-action selection is critical for learning the grasp-object relation. This identified that transition with non-optimal actions should be avoided to prevent getting stuck at a local minimum. Configuring the camera setup showed that Perceiver-Actor's voxel framework yields the highest success of handovers when obtaining great scene coverage. Furthermore, inspecting object properties showed that the grasp opportunity width and volume correlate to the action prediction error and failure rates, respectively. An increased volume relates to higher prediction error, and an increasing grasp opportunity width relates to precision limitations of the discretized action output from Perceiver-Actor's voxel framework. Furthermore, the object orientations collected from training and used in evaluation produce a coverage metric that is in line with the data distribution. This showed that successes of a behavior cloning agent in handover tasks are limited by the input data-distribution/object orientation. Hence, applying voxel-based RGB augmentation does not improve overall success. Furthermore, the Perceiver-Actor framework produces a rounding error when converting to world coordinate frames. Despite 'compounding error' being a common issue for behavior cloning agents, the trained policy was often able to reach the object. Finally, the end-to-end framework prevents usage of the perception component, avoiding segmentation errors or viewpoint change. Nonetheless, the discretization has been shown as the largest issue yielding inaccuracies.

To recapitulate, an end-to-end behavior cloning was used that avoids requiring a perception module. The behavior cloning agent can learn features for the handover task by learning the grasp-object relation. However, the performance of this method is limited in generalization and precision. Limited generalization is due to the sparse training data, and accuracy is due to the voxel input-output framework.

References

[1] Sergey Levine et al. “End-to-end training of deep visuomotor policies”. In: *Journal of Machine Learning Research* 17.39 (2016), pp. 1–40.

[2] Ian Lenz, Honglak Lee, and Ashutosh Saxena. “Deep learning for detecting robotic grasps”. In: *The International Journal of Robotics Research* 34.4-5 (2015), pp. 705–724.

[3] Jeannette Bohg et al. “Data-driven grasp synthesis—a survey”. In: *IEEE Transactions on robotics* 30.2 (2013), pp. 289–309.

[4] R L Lab. *Cornell grasping dataset*. 2013. URL: `http://pr.cs.cornell.edu/grasping/%20rect%20data/data.php`.

[5] Amaury Depierre, Emmanuel Dellandréa, and Liming Chen. “Jacquard: A large scale dataset for robotic grasp detection”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 3511–3516.

[6] Fu-Jen Chu, Ruinian Xu, and Patricio A Vela. “Real-world multiobject, multigrasp detection”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3355–3362.

[7] Douglas Morrison, Peter Corke, and Jürgen Leitner. “Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach”. In: *Proc. of Robotics: Science and Systems (RSS)*. 2018.

[8] Stefan Ainetter and Friedrich Fraundorfer. “End-to-end trainable deep neural network for robotic grasp detection and semantic segmentation from rgb”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 13452–13458.

[9] Valerija Holomjova, Andrew J Starkey, and Pascal Meißner. “GSMR-CNN: An End-to-End Trainable Architecture for Grasping Target Objects from Multi-Object Scenes”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 3808–3814.

[10] Martin Sundermeyer et al. “Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 13438–13444.

[11] Peiyuan Ni et al. “Pointnet++ grasping: Learning an end-to-end spatial grasp generation algorithm from sparse point clouds”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 3619–3625.

[12] Binglei Zhao et al. “Regnet: Region-based grasp network for end-to-end grasp detection in point clouds”. In: *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2021, pp. 13474–13480.

[13] Kaichun Mo et al. “Where2act: From pixels to actions for articulated 3d objects”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6813–6823.

[14] Ruihai Wu et al. “VAT-mart: Learning visual action trajectory proposals for manipulating 3d ARTiculated objects”. In: *arXiv preprint arXiv:2106.14440* (2021).

[15] Yiran Geng et al. “End-to-End Affordance Learning for Robotic Manipulation”. In: *arXiv preprint arXiv:2209.12941* (2022).

[16] Fanbo Xiang et al. “Sapien: A simulated part-based interactive environment”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11097–11107.

[17] Sergey Levine et al. “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection”. In: *The International journal of robotics research* 37.4-5 (2018), pp. 421–436.

[18] Patrick Rosenberger et al. “Object-independent human-to-robot handovers using real time robotic vision”. In: *IEEE Robotics and Automation Letters* 6.1 (2020), pp. 17–23.

[19] Lirui Wang et al. “Goal-auxiliary actor-critic for 6d robotic grasping with point clouds”. In: *Conference on Robot Learning*. PMLR. 2022, pp. 70–80.

[20] Natalie Sebanz, Harold Bekkering, and Günther Knoblich. “Joint action: bodies and minds moving together”. In: *Trends in cognitive sciences* 10.2 (2006), pp. 70–76.

[21] Valerio Ortenzi et al. “Object handovers: a review for robotics”. In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1855–1873.

[22] Yuke Zhu et al. “Reinforcement and imitation learning for diverse visuomotor skills”. In: *arXiv preprint arXiv:1802.09564* (2018).

[23] Corey Lynch and Pierre Sermanet. “Language conditioned imitation learning over unstructured data”. In: *arXiv preprint arXiv:2005.07648* (2020).

[24] Dylan J Foster, Adam Block, and Dipendra Misra. “Is behavior cloning all you need? understanding horizon in imitation learning”. In: *arXiv preprint arXiv:2407.15007* (2024).

[25] Yu-Wei Chao et al. “Handoversim: A simulation framework and benchmark for human-to-robot object handovers”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 6941–6947.

[26] Lirui Wang, Yu Xiang, and Dieter Fox. “Manipulation trajectory optimization with online grasp synthesis and selection”. In: *arXiv preprint arXiv:1911.10280* (2019).

[27] Wei Yang et al. “Reactive human-to-robot handovers of arbitrary objects”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 3118–3124.

[28] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. “A billion ways to grasp: An evaluation of grasp sampling schemes on a dense, physics-based grasp data set”. In: *The International Symposium of Robotics Research*. Springer. 2019, pp. 890–905.

[29] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. `http://pybullet.org`. 2019.

[30] Sammy Christen et al. “Learning human-to-robot handovers from point clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 9654–9664.

[31] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. “Perceiver-actor: A multi-task transformer for robotic manipulation”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 785–799.

[32] Andrew Jaegle et al. “Perceiver io: A general architecture for structured inputs & outputs”. In: *arXiv preprint arXiv:2107.14795* (2021).

[33] Edward Johns. “Coarse-to-fine imitation learning: Robot manipulation from a single demonstration”. In: *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2021, pp. 4613–4619.

[34] Yu-Wei Chao et al. “DexYCB: A benchmark for capturing hand grasping of objects”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 9044–9053.

[35] Stephen James and Andrew J Davison. “Q-attention: Enabling efficient learning for vision-based robotic manipulation”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 1612–1619.

[36] Snehal Jauhri, Ishikaa Lunawat, and Georgia Chalvatzaki. “Learning any-view 6dof robotic grasping in cluttered scenes via neural surface rendering”. In: *arXiv preprint arXiv:2306.07392* (2023).

[37] Berk Calli et al. “The ycb object and model set: Towards common benchmarks for manipulation research”. In: *2015 international conference on advanced robotics (ICAR)*. IEEE. 2015, pp. 510–517.

[38] Yu Xiang et al. “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes”. In: *arXiv preprint arXiv:1711.00199* (2017).

[39] Josh Tobin et al. “Domain randomization for transferring deep neural networks from simulation to the real world”. In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[40] TorchVision maintainers and contributors. *TorchVision: PyTorch’s Computer Vision library*. Nov. 2016. URL: `https://github.com/pytorch/vision`.

[41] Douglas Morrison, Peter Corke, and Jürgen Leitner. “Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation”. In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4368–4375.

[42] Sreenivas R Sukumar et al. “Towards understanding what makes 3D objects appear simple or complex”. In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE. 2008, pp. 1–8.

[43] Xiaoxiao Guo et al. “Hybrid reinforcement learning with expert state sequences”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 3739–3746.

[44] Hamidreza Kasaei and Mohammadreza Kasaei. “Mvgrasp: Real-time multi-view 3d object grasping in highly cluttered environments”. In: *Robotics and Autonomous Systems* 160 (2023), p. 104313.

[45] Stephen James et al. “Rlbench: The robot learning benchmark & learning environment”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3019–3026.

[46] Qiyu Dai et al. “GraspNeRF: Multiview-based 6-DoF Grasp Detection for Transparent and Specular Objects Using Generalizable NeRF”. In: (2023).

[47] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. “6-dof graspnet: Variational grasp generation for object manipulation”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 2901–2910.

[48] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. “Data quality in imitation learning”. In: *Advances in neural information processing systems* 36 (2023), pp. 80375–80395.

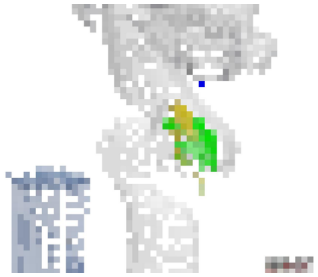
[49] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. “A reduction of imitation learning and structured prediction to no-regret online learning”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 627–635.

[50] Zifan Wang et al. “GenH2R: Learning Generalizable Human-to-Robot Handover via Scalable Simulation Demonstration and Imitation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 16362–16372.

[51] Sammy Christen et al. “Synh2r: Synthesizing hand-object motions for learning human-to-robot handovers”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 3168–3175.

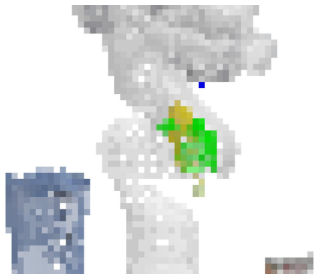
[52] Pete Florence et al. “Implicit behavioral cloning”. In: *Conference on robot learning*. PMLR. 2022, pp. 158–168.

A. Multi-camera setup: qualitative results

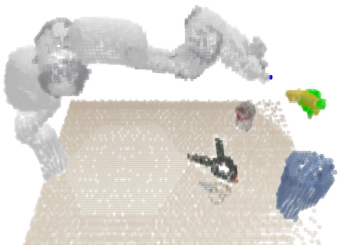


(a) Camera setup 1: front angle perspective

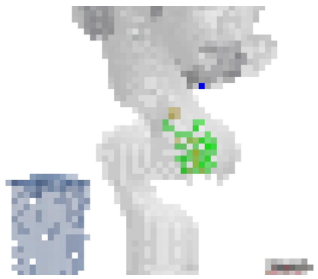
(b) Camera setup 1: side angle perspective



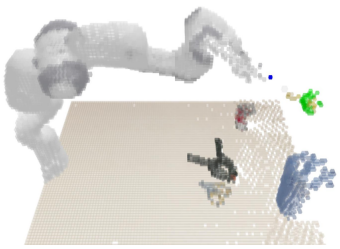
(c) Camera setup 2: front angle perspective



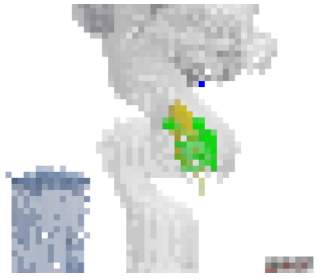
(d) Camera setup 2: side angle perspective



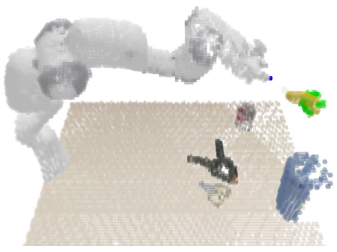
(e) Camera setup 3: front angle perspective



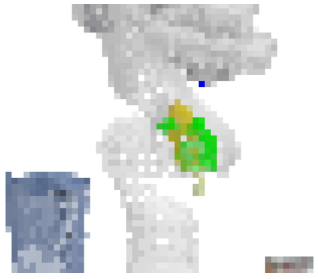
(f) Camera setup 3: side angle perspective



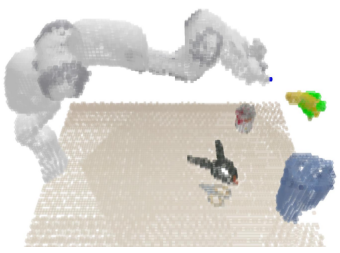
(g) Camera setup 4: front angle perspective



(h) Camera setup 4: side angle perspective



(i) Camera setup 5: front angle perspective



(j) Camera setup 5: side angle perspective

Figure A.1: Voxel observation for each camera setup.

B. RGB augmentation results

Object	summary				with grasp			without grasp			RMSE$_{grasp}$	
	success %	failure %			failure %			failure %			$\Delta_{distance}$ (m)	Δ_{GT} (m)
		contact	drop	timeout	contact	drop	timeout	contact	drop	timeout		
banana	10	20	30	40	10	10	40	10	20	0	0.064	0.024
bleach cleanser	10	10	80	0	0	30	0	10	50	0	0.085	0.065
bowl	40	10	30	20	0	0	20	10	30	0	0.079	0.06
cracker box	0	20	70	10	0	0	10	20	70	0	0.087	0.079
extra large clamp	0	0	50	50	0	10	50	0	40	0	0.161	0.151
foam brick	20	40	10	30	10	10	30	30	0	0	0.061	0.02
gelatin box	10	20	40	30	0	40	30	20	0	0	0.073	0.035
large marker	30	20	0	50	10	0	50	10	0	0	0.066	0.025
mug	10	10	30	50	0	0	40	10	30	10	0.066	0.026
mustard bottle	0	30	60	10	0	0	10	30	60	0	0.054	0.036
pitcher base	40	0	40	20	0	20	0	0	20	20	0.192	0.15
potted meat can	10	30	60	0	20	50	0	10	10	0	0.08	0.051
power drill	20	10	60	10	0	20	10	10	40	0	0.061	0.029
pudding box	40	10	30	20	0	20	10	10	10	10	0.066	0.025
scissors	10	0	0	90	0	0	90	0	0	0	0.116	0.086
sugar box	20	0	60	20	0	10	20	0	50	0	0.09	0.085
tomato soup can	30	0	70	0	0	30	0	0	40	0	0.063	0.023
tuna fish can	40	10	40	10	0	20	0	10	20	10	0.043	0.017

Table B.1: Success and failure rate for every object.

Object	summary				with grasp			without grasp			RMSE$_{grasp}$	
	success %	failure %			failure %			failure %			$\Delta_{distance}$ (m)	Δ_{GT} (m)
		contact	drop	timeout	contact	drop	timeout	contact	drop	timeout		
banana	0	30	40	30	10	30	30	20	10	0	0.1	0.069
bleach cleanser	10	20	60	10	10	0	10	10	60	0	0.08	0.063
bowl	70	0	10	20	0	10	20	0	0	0	0.117	0.083
cracker box	20	10	50	20	0	30	20	10	20	0	0.102	0.095
extra large clamp	0	10	30	60	0	0	50	10	30	10	0.102	0.091
foam brick	20	20	60	0	10	30	0	10	30	0	0.062	0.021
gelatin box	30	10	50	10	0	10	10	10	40	0	0.06	0.023
large marker	40	0	30	30	0	10	30	0	20	0	0.057	0.022
mug	10	0	40	50	0	0	50	0	40	0	0.064	0.032
mustard bottle	10	10	70	10	0	10	0	10	60	10	0.057	0.042
pitcher base	10	0	80	10	0	50	0	0	30	10	0.203	0.166
potted meat can	0	30	60	10	20	30	10	10	30	0	0.07	0.05
power drill	10	10	70	10	0	30	10	10	40	0	0.044	0.027
pudding box	20	20	30	30	10	10	20	10	20	10	0.062	0.047
scissors	0	0	30	70	0	10	70	0	20	0	0.067	0.041
sugar box	30	0	60	10	0	20	10	0	40	0	0.079	0.076
tomato soup can	30	0	60	10	0	50	0	0	10	10	0.067	0.024
tuna fish can	20	40	30	10	20	30	10	20	0	0	0.046	0.022

Table B.2: Success and failure rate for every object.