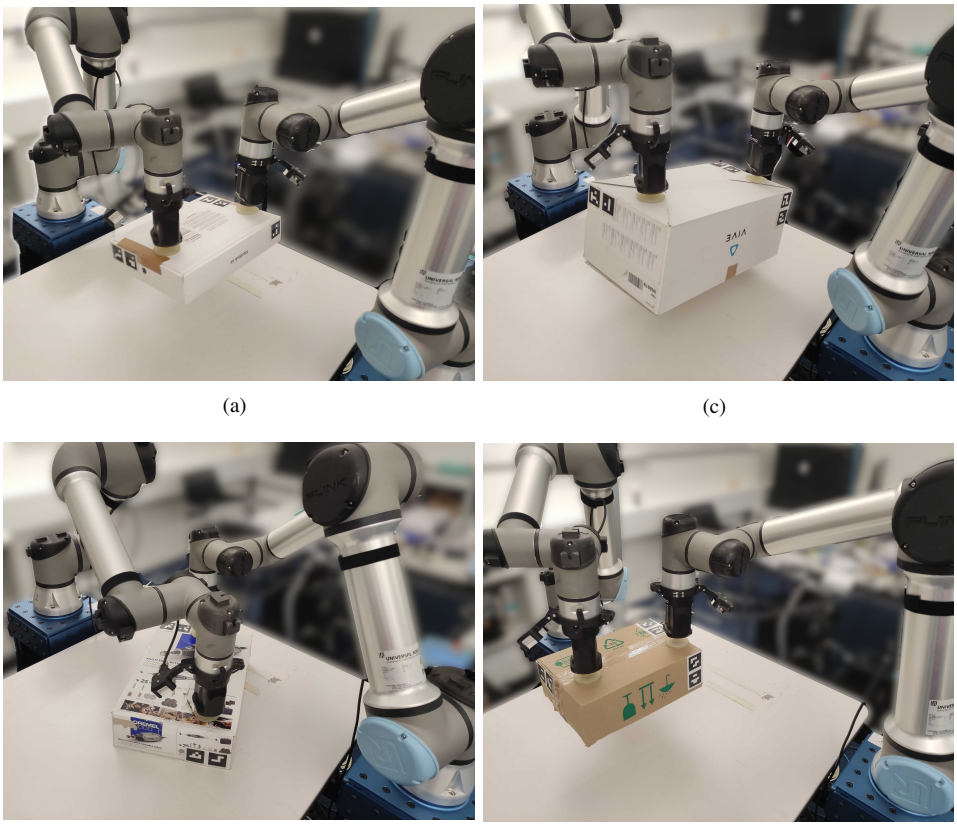
Two Arms, One Goal: Learning Dual-Robot Manipulation for Material Handling with Online Reinforcement Learning

Sebastiano Oliani, Valentin N. Hartmann, and Stelian Coros



(a) (c)

(b) (d)

Fig. 1: The dual-arm setup we used to collaboratively accomplish manipulation tasks with boxes characterized by different weights and sizes. Specifically, (a): lift, (b): rotation, (c): lift & rotation, (d): motion.

***Abstract*— In this work, we present a framework for dual-arm robotic systems actuated with suction cups manipulating boxes of different shapes and dimensions. We used an online reinforcement learning (RL) approach to investigate the capabilities of this setup and identify possible challenges, solutions, and applications. The setup integrates a perception system and visual augmentation to interpret the surroundings of the end effectors. RL allows simplifying the motion planning problem, particularly concerning collisions when the tasks are repetitive enough and the surrounding environment is well known. Using online RL approaches bootstrapped with human-expert demonstrations allows obtaining policies with little training time that accomplish the desired task while directly closing the sim-to-real gap. We introduced human interventions, Hindsight Experience Replay, and Residual Reinforcement Learning to improve the overall training effectiveness and efficiency. Considering the tasks faced more commonly in logistic centers, we trained policies that can grasp, lift, rotate and move boxes inside the workspace using two robots acting simultaneously on the objects. We validate our method through detailed real-world experiments. The code is available at https://github.com/sebastianooliani/dual-serl.**

All the authors are with the Computational Robotics Lab (CRL), ETH Zurich, CH.

Corresponding author: soliani@ethz.ch

I. INTRODUCTION

In modern industrial environments, the ability to automate complex manipulation tasks such as grasping, lifting, and positioning objects of varying shapes and sizes is crucial. Many factories use conveyor systems to transport objects for sorting, packaging, or assembly, and robotics can optimize these processes by improving speed and efficiency.

Designing an effective controller for dual-arm manipulation is a long-standing challenge. The problem was initially tackled using model-based approaches [1], [2], [3]. Those methods struggle when manipulating objects with significantly different sizes, weight distribution, and shape inconsistencies. The complexity grows even more when two

robots collaborate to simultaneously manipulate an object. The recent successes of Reinforcement Learning and Imitation Learning approaches incentivize robotics researchers to apply them to this problem [4], [5], [6], [7]. Even though manipulating a box might look trivial, there are not many solutions that actually use RL and are tested on real-world setups. Without properly designed reward functions, the agent is amenable to exploiting niches of the environment that create inappropriate, non-robust, and potentially unsafe solutions.

In this work, we compare the performance of learning-based policies for dual-arm manipulation of boxes to traditional model-based approaches. Policies for similar tasks were extensively studied for single-arm and dual-arm systems in simulation [8], [9], [10], [11], but only few real-world implementations are available [12], [13], [14]. To the best of our knowledge, no previous work trained an online RL policy for a dual-arm setup actuated with suction cups.

This work is strongly based on [15]. This study focuses on determining how different encoders can be used in a reinforcement learning framework to interpret the spatial environment in the local surroundings of a robot arm. It is the first work to train a reinforcement learning policy for a suction gripper using point cloud inputs with real-world data. In particular, it shows that 3D voxel grid representations are particularly beneficial for spatial perception, outperforming other image-based policies. As a result, point cloud is directly used in this work as the main visual backbone, without comparing its performance with other image-based encoders. The whole framework is also based on SERL [16], a software suite for robotic RL with sample-efficient off-policy algorithms. It provides a set of useful packages and libraries to control and train RL agents. In particular, training is performed online using several state-of-the-art algorithms such as Soft Actor Critic (SAC) [17], and Reinforcement Learning with Prior Data (RLPD) [18], and can be bootstrapped with human-expert demonstration.

To the best of our knowledge, our system is the first to (1) achieve dual-arm coordination with point cloud inputs using RL in real-world settings, (2) train policies for dual-arm systems using a limited number of human-expert demonstrations and within a relatively short time, and (3) train dual-arm manipulation policies actuated with suction grippers using Hindsight Experience Replay (HER) [19] and Residual RL (RRL) [20] while relying on data collected solely in the real-world.

II. RELATED WORK

The work in [2] proposes and validates a model predictive control (MPC) algorithm for online time-optimal trajectory planning of cooperative robotic manipulators. In particular, it shows that using two robot arms in a tight, shared workspace leads to an improvement in cycle times compared to deploying a single robot to execute the same tasks.

The work in [4] offers an overview and comparison of the main state-of-the-art algorithms for Imitation Learning in robotic manipulation. A successful approach based on Imitation Learning for bimanual manipulation is presented in [21]. It trains and deploys policies on low-cost robots, thanks to its Action-Chucking Transformer (ACT). The work in [12] introduced an Imitation Learning based method to deploy policies in dual-arm setups using a single demonstration.

The works presented in [9], [10], and [11] showed in simulation that the combination of RL with HER or its modification can effectively boost the performance of common manipulation tasks, like pushing, pick & place, and path following. As an example of a real-world implementation of HER, the work in [22] trained an RL policy that integrated HER in simulation and implemented it in a real-world setting for robot navigation.

The work in [14] deployed a dual-arm RL policy in the real-world for grasping large-sized objects. The policy was trained in simulation using a CNN-based Proximal Policy Optimization (PPO) algorithm [23] and used RGB images as visual backbone.

The work most similar to ours is [7]. It presents the first robotic system which achieves dual-arm coordination with image inputs using Reinforcement Learning in real-world settings. In particular, it bootstraps the training using human-expert demonstration and introduces a human-in-the-loop (HIL) feedback approach during the training. This work outperforms Imitation Learning algorithm methods that are trained on the same amount of human data to perform dual-arm tasks like object handover and collaborative assembly.

III. BACKGROUND

We formulate the general RL task as a partially observable Markov decision process (POMDP) [24], [25]. A POMDP can be described as a tuple $(\mathcal{O}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{O}$ is the observation space, $\mathcal{A}$ is the action space, p is the unknown and potentially stochastic transition probability that depends on the system dynamics, $r : \mathcal{O} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function that maps the current observation and action to reward $r_t = r(o_t, a_t)$ and γ is the discount factor. The goal is to find an optimal policy $\pi(a_t|o_t)$, that maximizes the expected cumulative return given by $\mathbb{E}_\pi \sum_{t=0}^{\infty} \gamma^t r_t$.

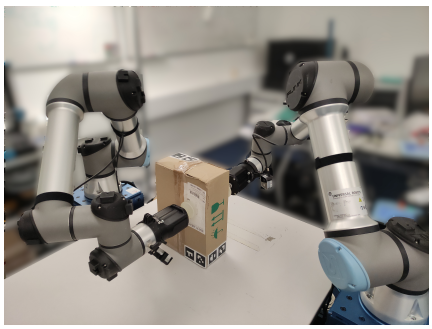
A. Soft Actor Critic (SAC)

SAC [17] is an RL algorithm for continuous control tasks that optimizes a stochastic policy π_θ by maximazing both policy entropy [26] and expected reward via the state-action value function Q_θ, encouraging exploration. Since SAC cannot effectively handle high-dimensional observation spaces like images or pointcloud data, we use this state-of-the-art off-policy algorithm as a baseline for comparison with more advanced policies.

B. Reinforcement Learning with Prior Data (RLPD)

RLPD [18] is an RL algorithm based on SAC, which aims to make real-world RL efficient. Key modifications to SAC include: (1) high update-to-data ratio during training, (2) symmetric sampling of prior and on-policy data, and (3) layer-norm regularization during training. While this method can train from scratch, it is often useful to bootstrap learning using prior data, for example demonstrations.

(a)



(b)

Fig. 2: In (a), two robots in the *top-grasping* configuration. In (b), the *side-grasping* configuration.

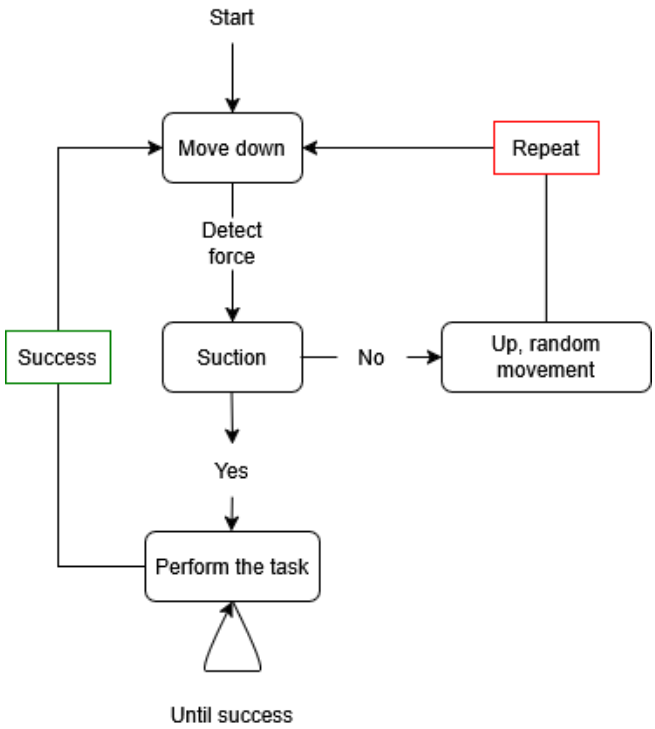


Fig. 3: A graph representing a general scheme of a Behavior Tree.

C. Data-Regularized Q (DRQ)

DRQ [27] is a simple data augmentation technique that can be applied to any standard model-free RL algorithm. In SERL, it is built on RLPD [18]. With this approach, we are able to find an optimal policy for a POMDP $(\mathcal{O}, \mathcal{A}, p, r, \gamma)$, using either RGB images, depth images or a voxel grid as additional observation inputs.

IV. METHODOLOGY

In this section, we describe the different methods we exploited to train RL policies that could be effectively deployed on dual-arm systems. Considering the tasks we want to achieve and starting from a simple model-based controller, we tackled the main challenges caused by the high dimensionality of the problem and the need for policies that require reasonable training time to converge.

A. Creating a benchmark for dual-arm manipulation

Creating a benchmark for dual-arm manipulation requires selecting possible tasks that robots must perform. Considering the necessities faced more commonly in logistics centers, we trained policies to:

1) lift a box,
2) rotate a box around one of its axes,
3) combine a lift and a rotation of a box,
4) move a box to different positions inside the workspace.

The differences in the task complexities require different approaches that are covered in the following paragraphs (Sec. IV-B, Sec. IV-C, Sec. IV-D, Sec. IV-E). A more detailed definition of every policy can be found in Sec. V-A, Sec. V-B, Sec. V-C, and Sec. V-D.

The first problem we faced was ensuring the two robots could grasp the object securely without letting it fall. Initially, two grasping strategies were taken into account (see Fig. 2). A so-called *top-grasping* strategy, with the end effectors facing the table surface, and a *side-grasping* strategy, where the end effectors are pointing at each other. As one might expect, the choice of the grasping strategy depends on the boxes' dimensions, particularly on which plane lies the longest dimension.

The first idea is to compare the performance of a simple controller against policies trained with Reinforcement Learning. This first controller allowed us to identify possible failure cases that can be fixed by exploiting learning-based approaches.

For example, we created a controller based on a behavior tree (BT) to perform the desired tasks and establish the first baselines (see Fig. 3). It is straightforward to realize that policy performance may vary. We observed that suction cups struggle when grasping boxes that are covered with tape or that present holes on their surface (see Fig. 4). Small changes in the box's initial position and orientation have a huge influence, too.

For the sake of simplicity and considering the relative positions of the robots, in this work, we considered only the *top-grasping* strategy.

Fig. 4: Possible failures in the grasping of a box. Holes, tape, and broken parts make grasping challenging.

B. Integrating visual information

Visual information is fed into the policy's neural network to provide local information about the box. It was shown that these enhance the adaptability of the policy to different boxes [27]. Several visual backbones can be adopted in SERL, including, for example, grey or RGB images with ResNet. Voxel-SERL integrates a voxel-grids-based method, which showed an increase in performance over the previous methods [15]. The visual inputs are taken from a camera mounted on each wrist of the robot and then processed by a neural network. Training lasts longer when visual information is added to the observation space of the policy. While running the experiments, we noticed that RLPD tends to explore more than a simple SAC algorithm, taking more time to converge. We realized that this exploration originated issues with collisions between the robots and singularity configurations, which were not encountered during the training without wrist camera information.

Therefore, the system must frequently check if it is in a collision or singularity configuration. The former is evaluated by computing the relative 2-norm distance between the tip of the gripper, the end-effectors, and the 5th and the 4th joints of the two robots, the latter is avoided by shrinking the workspace of the robots.

In order to avoid actions that move the robots into a collision or a singularity, during training we penalized the agent if the robots were getting too close, leading to the end of the episode too. In addition, a custom reset procedure is needed to avoid collisions once the end of the episode is reached. When training policies using RLPD, we introduced an additional cost on the relative end-effector distance.

C. Introducing human interventions

The complexity required to train reinforcement learning policies is closely linked to the cardinality of the state and action spaces, as well as the task horizon, assuming an appropriate exploration policy ([28], [29], [30], [31]). Increases in the size of the state/action spaces, task horizon, or their combination lead to a proportional rise in the number of samples required to learn an optimal policy. This can eventually make online reinforcement learning impractical. Those challenges can be overcome by allowing a human operator to guide the learning process by intervening at random steps during training. This is what [7] refers to as **human-in-the-loop feedback**.

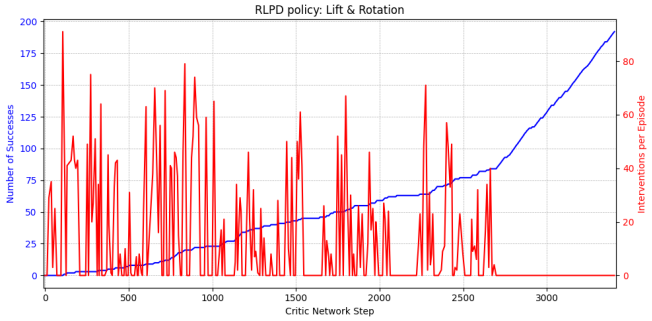


Fig. 5: Number of human interventions per episode compared to the number of successes performed during the training of the RLPD policy that performs a lift combined with a rotation of the box.

More in detail, a human operator supervises the robot during training and provides corrective actions to explore more efficiently. Interventions can be easily performed by acting on the SpaceMouse. When a human intervenes, the action read from the SpaceMouse is applied to the robot instead of the policy's action.

Unlike the implementation available in [7], one single replay buffer is used in all tasks except the motion. In those tasks, the policy's transitions are added to the buffer together with the interventions.

Using human interventions can massively speed up the training, but it comes with some negative drawbacks too. As can be quickly realized empirically, providing long and sparse interventions that lead to task success will cause the overestimation of the value function. This happens in particular when interventions are frequent in the early stages of the training process. These result in unstable training dynamics.

In our experience, the policy improves faster when the human operator issues specific corrections while letting the robot explore on its own otherwise. In Fig. 5, one can visualize the number of interventions per episode performed together with the number of successes during the training of a policy to lift and rotate a box. Once the policy becomes good enough, the number of successes grows even without human interventions.

Considering the lifting task as an example, intervening just to correct wrong behaviors while learning to grasp or lift keeps the training stable and guides the exploration towards a correct and successful policy.

Algorithm 1: Integrating HER in the training loop

```
Initialize demo buffer 𝒟
Replay buffer ℛ ← ∅
Sample a goal g
Transitions trans ← {}
HER Transitions her_trans ← {}
for episode=1,M do
    for t=1,N do
        trans ← {s_t, a, r(s_t, a, g), s_{t+1}};
        if episode finishes then
            g_her ← p^box;            /* Define new goal */
            if episode is successful then
                ℛ ← {trans};
            else
                her_trans ← {s_t, a, r(s_t, a, g_her), s_{t+1}};
                 /* Recompute transitions */
                if new episode is successful then
                    ℛ ← {trans};
                    ℛ ← {her_trans};
                else
                    ℛ ← {trans};
                end
            end
        end
    end
    trans ← {};
    her_trans ← {};
end
```

D. Hindsight Experience Replay (HER)

Considering the motion planning task, we needed a smart strategy to record demonstrations to bootstrap the training of the reinforcement learning policy. There are two main problems: knowing the goal position before actually reaching it, and moving the box toward this location. To simplify this step, we implemented **Hindsight Experience Replay** [19]. In this paper, HER is implemented to train control policies in a simulation that accomplish three manipulation tasks: pushing, sliding, and pick-and-place. Another work used HER together with demonstrations in a single-arm setup to solve a block stacking task [8]. Similarly to our scenario, the replay buffer was initialized with 100 human-expert demonstrations, but the final policy was only trained and evaluated in simulation. HER was integrated only during training to profit from the experience gained from unsuccessful episodes.

During the recording, one can sample a goal position inside the dual-robot workspace. In the beginning, this goal position is used as a *fake* goal. This means that we do not know exactly where the goal is, but we still compute rewards and observations based on it. So, the boxes are just moved randomly inside the workspace. At the end of the recording of every demonstration, the final box position is considered as the *real* goal position, and all the rewards and observations are recomputed according to this goal.

At the end of all the unsuccessful episodes, we collect two sets of transitions. One in which the goal is not actually reached and one in which the goal is achieved. The latter are saved in the replay buffer and used to bootstrap the training. The same post-processing strategy is used at training time, as explained in Algorithm 1. When training the policies, two sets of transitions are saved in the replay buffer only when the episode ends unsuccessfully, but the box was grasped at least in the last step. This choice aimed to reduce the number of unsuccessful transitions in the replay buffer, in particular during the early steps of the training. Hindsight Experience Replay is not needed when the box is not grasped or the episode ends successfully.

TABLE I: The dimension of the boxes used during training and evaluation.

Box ID	X [mm]	Y [mm]	Z [mm]	Group
50	342	452	130	Seen
51	245	335	72	Unseen
52	268	391	102	Unseen
53	249	431	189	Unseen
54	221	310	105	Unseen

E. Residual Reinforcement Learning (RRL)

Residual Reinforcement Learning [20] represents a variation of traditional learning-based techniques. To take advantage of the efficiency of conventional controllers but also the flexibility of RL, the control action is defined as:

$$u = \pi_H(s) + \pi_\theta(s) \tag{1}$$

where $\pi_H(s)$ is a human-designed feedback control law and $\pi_\theta(s)$ is a learned policy parametrized by θ and optimized by an RL algorithm to maximize the expected returns on the task. This approach simplifies the training of the secondary policy and enables its use as an optimizer for the primary one. Other works in manipulation substitute the human-designed feedback control law with a BC policy [32].

In this work, we integrated RRL only in the motion task to guide the robotic arms toward the desired goal position. The control action applied to the robots is computed as follows:

$$u_t = (a_{FF_t} + a_t) \cdot k \tag{2}$$

where a_t is the action at time t, a_{FF_t} is the feed-forward action, which in our case is the normalized vector representing the direction that connects the goal position to the actual box position, and k is the scaling factor. When recording the demonstrations, a_t is the SpaceMouse command, while during training, it is the output of the actor network.

V. EXPERIMENTS

For the experiments, we used two 6-DoF UR5e Robot Arms, each one with the Robotiq EPick vacuum gripper. The demonstrations were recorded using two separate SpaceMouses. On the wrist of the end-effectors, we placed an Intel RealSense D405 camera for each arm to augment the vision capabilities. Additionally, an Orbbec Gemini RGBD camera is used for the pose estimation of the boxes (see Fig. 6). This camera is only used in a separate thread to detect the position of the box via ArUco tags [33].

A simple behavior tree is used as a baseline for every task. Policy training was done using just one box (ID=50). Each RL-based policy was initialized with 20 successfully

(a) (b)

Fig. 6: In (a), the setup used for the experiments. In (b), the boxes used during policy evaluation.

teleoperated demonstrations, except one of the motion planning policies, which was initialized with 60 demonstrations, as we will discuss in Sec. V-D.

The policies' performance must be evaluated in different scenarios while handling boxes of different dimensions, weights and shapes. These boxes are visualizable in Fig. 6. The sizes of the boxes are reported in Table I. It is worth mentioning that some physical properties of the boxes can be unknown or hard to estimate. No information about the weight of the box, the dimensions, the friction coefficient, or the location of the center of mass is considered as an input. For example, Box 50 contains several smaller boxes inside itself, so the location of the center of mass is variable over time and across different episodes.

A pre-trained VoxNet is adopted as a visual backbone to augment the observation spaces when running the RLPD algorithm.

A brief description of each task can be found in the Sec. V-A, Sec. V-B, Sec. V-C, and Sec. V-D. All the observations and rewards are reported in the appendix (Sec. VI). We note that all the angles are represented using Modified Rodrigues Parameters. The evaluation metrics are (1) **the success rate**, which is 1 for a successful episode and 0 for a failure with no partial credits; (2) **a set of sub-successes** that characterize every task; (3) the **time** it takes for a trajectory to reach the goal state or time out.

A. Lift policy

The first task is to lift a box by 5 cm. In more detail, the success condition is achieved when both robots are grasping the box, and the actual robot's TCP positions are 5 cm higher than the robot's initial TCP positions.

This task is the only one that does not integrate the box position into the observations. The observation space, the reward weights, and the success condition are reported in Table VIII and Table IX.

B. Reorientation policy

The second task is to rotate a box around its z-axis. The initial orientation of the box can be estimated using the vision system. Then, a $40°$ rotation of the box is achieved using the *top-grasping* configuration. The observation space, the reward weights, and the success condition must be changed accordingly, as shown in Table XI, Table XII and Table XIII. More in detail, a positive reward is given for an anti-clockwise rotation of the box. When tuning the rewards, we found that computing the relative rotation using angles expressed in angle-axis representation improved the results. The success condition is achieved when both robots are grasping the box, and the box has rotated $40°$ around the z-axis from its initial position.

C. Combining lift and reorientation

The third task is a combination of the two previous tasks, i.e. lifting the box for about 5 cm and rotating it around one axis at a random angle (sampled between $15°$ and $25°$ at the beginning of every episode). The observation space, the reward weights, and the success condition must be changed accordingly, as shown in Table XIV, Table XV, Table XVI. The success condition is achieved when both robots are grasping the box, and the box has been correctly lifted and rotated from its initial position.

D. Motion Planning

A more general implementation does not rely solely on a simple lifting policy but includes a more general set of achievable motions. That is, an agent capable of moving an object to a desired position or orientation in the workspace. Therefore, the third task is to move a box to reach different goal positions within the workspace. The success condition must be designed to adapt to different goal positions [34]. In this work, we simplified the sampling of the goal position to a smaller region of space, i.e. a plane. This decision simplified

TABLE II: Policy specifications, training times and conditions.

Task	*ID*	*Policy*	*Visual Backbone*	*Pose Estimation*	*Training Time*	*HIL*	*Demos number*
Lift	(1)	BT	✗	No	✗	No	✗
	(2)	SAC	✗	No	12 minutes	No	20
	(3)	RLPD	VoxNet	No	50 minutes	Yes	20
Reorientation	(4)	BT	✗	Yes	✗	No	✗
	(5)	SAC	✗	Yes	12 minutes	No	20
	(6)	RLPD	VoxNet	Yes	50 minutes	Yes	20
Lift & Rotation	(7)	BT	✗	Yes	✗	No	✗
	(8)	SAC	✗	Yes	12 minutes	Yes	20
	(9)	RLPD	VoxNet	Yes	90 minutes	Yes	20
Motion	(10)	BT	✗	Yes	✗	No	✗
	(11)	SAC	✗	Yes	40 minutes	Yes	60
	(12)	SAC + RRL	✗	Yes	52 minutes	Yes	20

the problem setup but can still give valuable insights into the performance of learning-based controllers in those scenarios.

We implemented *Hindsight Experience Replay* [19] as described in Sec. IV-D to bootstrap the training with human-expert demonstrations.

Similarly to other tasks, the observation space and the success condition must be changed accordingly, as shown in Table XVII, Table XIX, Table XVIII and Table XX. More in detail, the success condition is achieved when both robots are sucking and the distance between the goal position and the box position is less than 5 cm. Information about the orientation of the box is not taken into account. During the evaluation, we sampled 30 different goal positions to test the capabilities of the policies.

During the training of this policy, we observed that several elements improved the performance of the agent:

- removing *tcp_force*, *tcp_torque*, *tcp_pos_diff*, *joint_positions*, from the observations, as reported in Table XVII and Table XIX.
- giving a reward based on the projection of the actual box position on a line that starts from the initial box position and ends at the goal position. The formula of this reward is Eq. (3):

$$norm(\mathbf{p}_t^{box}, \mathbf{p}_{t-1}^{box}) = \frac{(\mathbf{p}_t^{box} - \mathbf{p}_{t-1}^{box}) \cdot (\mathbf{p}_{goal} - \mathbf{p}_0^{box})^2}{||\mathbf{p}_{goal} - \mathbf{p}_0^{box}||_2^2} \tag{3}$$

  where $\mathbf{p}_t^{box}$ is the position of the box at time t, $\mathbf{p}_{goal}$ is the sampled goal position, and $\mathbf{p}_0^{box}$ is the initial box position.
- using two replay buffers. One is initialized with demonstrations, and one is filled up with just online experience and human interventions. When computing the optimization steps, we equally sampled the batches from each buffer and then concatenated them.
- increasing the amount of initial demonstrations. This is beneficial because it increases the coverage of the points in the workspace.
- using RRL, which already gives a direction along which the end effectors should move. Then, the policy just needs to learn to grasp the box first and then move along this direction without letting it fall.

E. Evaluation details

We reported in Table II all the tasks and policies we evaluated, together with their main details. For the motion task, it was not possible to train a policy with the RLPD algorithm due to constraints in the computational power of the computer we used. Since this was realized toward the end of this work, we just trained policies based on SAC.

Taking inspiration from [35], we defined a set of sub-successes to capture all the details of the resulting policies and provide the reader with a deeper understanding of the capabilities of the system. For each of them, we computed the average values over the 30 episodes and over the number of boxes we used. To enhance readability, we reported the sub-successes in the Table XXI, Table XXII, Table XXIII, and Table XXIV in the appendix. Therefore, we defined:

- **grasp sub-success**: whether the box is grasped by one of the arms during the entire episode.
- **lift sub-success**: whether the box is lifted above the required threshold during the entire episode.
- **rotation sub-success**: whether the box is rotated above the required threshold during the entire episode.
- **motion sub-success**: whether the box reaches the desired goal position during the entire episode.
- **final distance from the goal**: the distance at the end of the episode from the goal position achieved during the entire episode.

To clarify, not all the sub-successes were used for every task, but each task has its characteristic sub-successes.

We summarized compactly the performance of the policies in Table III. Here, we reported the success rates of the

TABLE III: Policy evaluation. For every policy, 30 runs are performed.

Task	***Policy***	***Seen Box***		***Unseen Boxes***	
		Success Rate [%]	*Time [s]*	*Success Rate*	*Time [s]*
Lift	BT	77	**5.57** $\pm$ 2.54	**66**	**6.86** $\pm$ 2.20
	SAC	70	9.07 $\pm$ 5.14	28	11.79 $\pm$ 3.58
	RLPD	**100**	11.49 $\pm$ 3.00	55	12.88 $\pm$ 6.20
Rotation	BT	60	7.80 $\pm$ 1.92	62	**6.95** $\pm$ 1.91
	SAC	**100**	8.94 $\pm$ 0.59	75	10.99 $\pm$ 3.74
	RLPD	**100**	**5.70** $\pm$ 0.38	**90**	8.52 $\pm$ 5.92
Lift & Rotation	BT	60	8.71 $\pm$ 1.31	61	**8.38** $\pm$ 1.66
	SAC	**97**	7.71 $\pm$ 2.24	28	12.15 $\pm$ 3.86
	RLPD	**97**	**4.85** $\pm$ 2.25	**69**	8.49 $\pm$ 3.96
Motion	BT	67	**8.80** $\pm$ 1.23	**55**	**9.06** $\pm$ 1.03
	SAC	40	11.62 $\pm$ 3.2	30	12.93 $\pm$ 4.25
	SAC + RRL	**70**	11.67 $\pm$ 4.61	37	14.19 $\pm$ 3.03

policies on seen and unseen boxes, together with their average and standard deviation episode length.

As an important disclaimer, the results were collected over three months. Therefore, there could be minor differences in the evaluation. Due to time constraints, it was not possible to recollect all the data, but we can still compare the performances of the different policies and make valuable conclusions.

F. Results

Considering the first three tasks, i.e. lift, reorientation and their combination, the policies (3), (6), and (9) trained using point cloud as visual backbone perform better than the baselines (1), (4), (7) during the evaluation on the seen box. The success rate is greater than 97%, and the time required to complete the task is lower, except for (3). Considering SAC, the performances of (5) and (8) are better than the baseline on the seen box, except in (2), where they are worse than the baseline. Considering the motion task, only (12) achieved a success rate comparable to the baseline policy (10) on the seen box.

Considering only unseen boxes, the performance decreases in the success rate. Only (6) could maintain a success rate higher than 90%, while (9) performed slightly better than the baseline, achieving a 69% success rate. Anyway, the results still look more promising when evaluating the sub-successes we defined for every task. Only (8) and (11) performed always worse than the baseline. Notably, (12) achieved a higher motion sub-success rate than the baseline (10), and both (11) and (12) reduced the average final distance of the box from the goal at the end of the episode, which is, in the end, the main requirement for this task.

The success rate can be seen as a lower bound of these sub-successes. Due to the grasping condition on the success, it sometimes happened that the task was completed by using just one robot arm, as visible in Fig. 7.

In those cases, the task was not successful due to the definition of success. Nevertheless, it could still be the case that the box reached the desired final configuration, whether it was a rotation or a simple motion to a goal position. For example, as reported in Table XXI, (2) achieved a 28% success rate on the unseen boxes, but the lift sub-success for this task is 66% because just one arm grasped and lifted the box correctly. Similarly, as reported in Table XXII, (5) achieved 98% rotation sub-success on the unseen boxes, while the success rate is just 75%.

In conclusion, policies (5) and (6) trained for the reorientation task were the most successful on the unseen boxes among these three tasks.

G. Discussion

Introducing visual information improved the policies' success rates, especially in the unseen scenario. From the results, it is clear that using only discrete information, for example, relying only on the box position and/or orientation, is not sufficient to achieve a policy that is general enough to work for every box. This also comes with the cost of a minimum four times longer training time. Still, compared to other works, adding human interventions reduced the training time and was, most of the time, the key element to obtain a successful policy.

The worst performances on the unseen boxes can be related to the different sizes of the boxes, which have a big influence during the evaluation. For example, (8) and (9) struggled on the smaller boxes we used (ID 51 and 54), lowering the success rate of the whole task. We didn't introduce more boxes at training time due to the constraints of the setup, but this should definitely be done in the future to enhance the policy's success rate.

As we mentioned in Sec. V-E, we couldn't train a policy using the RLPD algorithm for the motion task. As we saw with the other tasks, using this algorithm together with point

(a)

(b)

Fig. 7: In (a), one single arm is lifting the box. In (b), one arm is rotating the box. Both policies achieve their sub-successes, respectively lifting and rotating, but do not grasp the box with both arms, leading to an unsuccessful episode.

cloud visual augmentation improves the performance of the policies. We expect a similar trend for the motion task, so it would be worth trying it out in future work.

While training a policy for the motion planning task, we found that using a similar pipeline as the one adopted for the previous tasks, i.e. initializing the replay buffer with 20 human-expert demonstrations, was not enough to obtain a successful policy. In our opinion, the main difference with the first three tasks is that training online a policy to move an object to a point in a portion of space requires covering many more possible cases than the ones needed to train the other policies. The actions across the different episodes might not be similar and repetitive enough, therefore making the optimization problem harder. One could simplify the problem setup to always reach the same position in the space, but then we would need to retrain a new policy for every possible position in the space.

Therefore, we considered three solutions to fix this problem. First, as in [7] and [8], we introduced two replay buffers. One is filled with demonstrations, and one with the online experience of the robots. During training, we equally

TABLE IV: Policy evaluation of the motion task. Results on box 50. 30 episodes are performed for every policy. The first policy is considered the baseline for the other two.

Task	*Policy*	*Success Rate [%]*	*Motion Sub-Success [%]*	*Demos*
Motion	SAC	3	13	20
	SAC	40 (+37)	53 (+40)	60
	SAC+RRL	**70** (+67)	**80** (+67)	20

TABLE V: Policy evaluation of the motion task. Trained with SAC using 60 human-expert demonstrations, but with different rewards and observations. For every policy, 30 runs are performed.

Task	*Policy*	*Success Rate [%]*		*Motion Sub-Success [%]*	
		Seen	*Unseen*	*Seen*	*Unseen*
Motion	SAC	40	**30**	53	**47**
	SAC	**67**	11	**70**	13

sampled the data from both replay buffers. In this way, we keep sampling and improving our policies using half of the data from the human-expert demonstrations recorded. Then, we trained two different policies. One in which we simply recorded more demonstrations, and another in which we kept the same number of demonstrations but introduced RRL as described in Sec. IV-E to act as a feedforward compensation term and help the policy to perform the required task.

As reported in Table IV, when evaluating the performance on box 50, both success rates grow with respect to the performance of the policy simply initialized with 20 demonstrations, respectively by 37% and 67%. Additionally, the motion sub-success improved respectively by 40% and 67%. We did not study how the number of initial demonstrations influences the performance, and this number was picked considering the time required to record them.

We performed several training runs to come up with a policy for the motion task, changing reward functions, weights, observations, or using the different techniques we mentioned in the previous paragraphs and sections. Therefore, we obtained different policies using different solutions. As reported in Table V, many policies performed slightly better than the one we reported in Table III, but had a lower success rate on the unseen boxes. As our goal is to find a policy that effectively manipulates the widest and diverse set of boxes, we picked as a decision criterion that the better policies are those that achieve higher success rates on the unseen boxes. Additionally, when running the evaluations, we noticed that different checkpoints of the network from the same training round have very different performances, as reported in Table VI. Due to time constraints, we did not study this phenomenon extensively, but we mentioned it as the results of the policies and the related considerations at different checkpoints differ considerably. This made the whole evaluation more challenging and time-consuming.

TABLE VI: Policy evaluation of the motion task. Trained with SAC using 60 human-expert demonstrations and evaluated at different checkpoints on box 50. For every policy, 30 runs are performed. The first policy is considered the baseline for the other two.

Task	*Policy*	*Checkpoint*	*Success Rate [%]*
Motion	SAC	15000	**67**
	SAC	20000	33 (-34)
	SAC	50000	30 (-37)

VI. CONCLUSIONS

To the best of our knowledge, our approach is the first to train a reinforcement learning policy for a dual-arm system actuated with suction cups, using point cloud inputs, integrating HER and RRL while relying only on real-world data. In this work, we exploited the capabilities of this setup in performing learning-based object manipulation tasks, comparing the results with behavior tree baselines.

In future work, we want to explore other techniques to train robust policies while keeping short training times. Online reinforcement learning policies require much less data than policies trained in simulation. This makes training more challenging when the goal of the task and, therefore, the actions are not repetitive and change across different episodes. Strategies to increase the sampling efficiency are needed in these cases.

Increasing the number of initial demonstrations showed a possible way to enhance performances, but at a non-negligible cost of time. For example, recording 60 successful motion planning demonstrations took around 20 minutes, thanks to the fact that HER was used to recompute unsuccessful trajectories.

One possible solution would be to transform the observation reference frames. For example, representing all the observations with respect to the reference frame of the box or with respect to the goal position. This solution was not implemented because the noise coming from the pose estimation could have ruined the performance. Additionally, to the best of our knowledge, the literature related to this topic is scarce.

As mentioned in Sec. V-F, introducing more boxes at training time would also be beneficial for the policies and better generalize the performances in unseen scenarios.

ACKNOWLEDGMENT

The authors thank Flink Robotics for practical and intellectual support.

REFERENCES

[1] "Dual arm robot for flexible and cooperative assembly". In: *CIRP Annals* 60.1 (2011), pp. 5–8. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2011.03.017. URL: https://www.sciencedirect.com/science/article/pii/S0007850611000187.

[2] Argtim Tika and Naim Bajcinca. "Predictive Control of Cooperative Robots Sharing Common Workspace". In: *IEEE Transactions on Control Systems Technology* 32.2 (2024), pp. 456–471. DOI: 10.1109/TCST.2023.3331525.

[3] Seyed Sina Mirrazavi Salehian, Nadia Figueroa, and Aude Billard. "A unified framework for coordinated multi-arm motion planning". In: *The International Journal of Robotics Research* 37.10 (2018), pp. 1205–1232. DOI: 10.1177/0278364918765952. eprint: https://doi.org/10.1177/0278364918765952. URL: https://doi.org/10.1177/0278364918765952.

[4] Yijiong Lin et al. *Bi-Touch: Bimanual Tactile Manipulation with Sim-to-Real Deep Reinforcement Learning*. 2023. arXiv: 2307.06423 [cs.RO]. URL: https://arxiv.org/abs/2307.06423.

[5] Satoshi Kataoka et al. *Bi-Manual Manipulation and Attachment via Sim-to-Real Reinforcement Learning*. 2022. arXiv: 2203.08277 [cs.RO]. URL: https://arxiv.org/abs/2203.08277.

[6] Michael Drolet et al. *A Comparison of Imitation Learning Algorithms for Bimanual Manipulation*. 2024. arXiv: 2408.06536 [cs.RO]. URL: https://arxiv.org/abs/2408.06536.

[7] Jianlan Luo et al. *Precise and Dexterous Robotic Manipulation via Human-in-the-Loop Reinforcement Learning*. 2024. arXiv: 2410.21845 [cs.RO]. URL: https://arxiv.org/abs/2410.21845.

[8] Ashvin Nair et al. *Overcoming Exploration in Reinforcement Learning with Demonstrations*. 2018. arXiv: 1709.10089 [cs.LG]. URL: https://arxiv.org/abs/1709.10089.

[9] Yuming Huang et al. *MRHER: Model-based Relay Hindsight Experience Replay for Sequential Object Manipulation Tasks with Sparse Rewards*. 2024. arXiv: 2306.16061 [cs.RO]. URL: https://arxiv.org/abs/2306.16061.

[10] Maria Makarova, Qian Liu, and Dzmitry Tsetserukou. *CONTHER: Human-Like Contextual Robot Learning via Hindsight Experience Replay and Transformers without Expert Demonstrations*. 2025. arXiv: 2503.15895 [cs.RO]. URL: https://arxiv.org/abs/2503.15895.

[11] Dániel Horváth et al. "HiER: Highlight Experience Replay for Boosting Off-Policy Reinforcement Learning Agents". In: *IEEE Access* 12 (2024), pp. 100102–100119. ISSN: 2169-3536. DOI: 10.1109/access.2024.3427012. URL: http://dx.doi.org/10.1109/ACCESS.2024.3427012.

[12] Yilong Wang and Edward Johns. *One-Shot Dual-Arm Imitation Learning*. 2025. arXiv: 2503.06831 [cs.RO]. URL: https://arxiv.org/abs/2503.06831.

[13] Yao Mu et al. *RoboTwin: Dual-Arm Robot Benchmark with Generative Digital Twins (early version)*. 2025.

arXiv: 2409.02920 [cs.RO]. URL: https://arxiv.org/abs/2409.02920.

[14] Yongliang Wang and Hamidreza Kasaei. *Learning Dual-Arm Coordination for Grasping Large Flat Objects*. 2025. arXiv: 2504.03500 [cs.RO]. URL: https://arxiv.org/abs/2504.03500.

[15] Nico Sutter, Valentin N. Hartmann, and Stelian Coros. *A comparison of visual representations for real-world reinforcement learning in the context of vacuum gripping*. 2025. arXiv: 2503.02405 [cs.RO]. URL: https://arxiv.org/abs/2503.02405.

[16] Jianlan Luo et al. *SERL: A Software Suite for Sample-Efficient Robotic Reinforcement Learning*. 2024. arXiv: 2401.16013 [cs.RO]. URL: https://arxiv.org/abs/2401.16013.

[17] Tuomas Haarnoja et al. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. 2018. arXiv: 1801.01290 [cs.LG]. URL: https://arxiv.org/abs/1801.01290.

[18] Philip J. Ball et al. *Efficient Online Reinforcement Learning with Offline Data*. 2023. arXiv: 2302.02948 [cs.LG]. URL: https://arxiv.org/abs/2302.02948.

[19] Marcin Andrychowicz et al. *Hindsight Experience Replay*. 2018. arXiv: 1707.01495 [cs.LG]. URL: https://arxiv.org/abs/1707.01495.

[20] Tobias Johannink et al. *Residual Reinforcement Learning for Robot Control*. 2018. arXiv: 1812.03201 [cs.RO]. URL: https://arxiv.org/abs/1812.03201.

[21] Tony Z. Zhao et al. *Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware*. 2023. arXiv: 2304.13705 [cs.RO]. URL: https://arxiv.org/abs/2304.13705.

[22] Shanze Wang et al. *MAER-Nav: Bidirectional Motion Learning Through Mirror-Augmented Experience Replay for Robot Navigation*. 2025. arXiv: 2503.23908 [cs.RO]. URL: https://arxiv.org/abs/2503.23908.

[23] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[24] RICHARD BELLMAN. "A Markovian Decision Process". In: *Journal of Mathematics and Mechanics* 6.5 (1957), pp. 679–684. ISSN: 00959057, 19435274. URL: http://www.jstor.org/stable/24900506 (visited on 04/02/2025).

[25] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial Intelligence* 101.1 (1998), pp. 99–134. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(98)00023-X. URL: https://www.sciencedirect.com/science/article/pii/S000437029800023X.

[26] Brian D. Ziebart et al. "Maximum entropy inverse reinforcement learning". In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*. AAAI'08. Chicago, Illinois: AAAI Press, 2008, pp. 1433–1438. ISBN: 9781577353683.

[27] Ilya Kostrikov, Denis Yarats, and Rob Fergus. *Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels*. 2021. arXiv: 2004.13649 [cs.LG]. URL: https://arxiv.org/abs/2004.13649.

[28] Chi Jin et al. *Is Q-learning Provably Efficient?* 2018. arXiv: 1807.03765 [cs.LG]. URL: https://arxiv.org/abs/1807.03765.

[29] Chi Jin et al. *Provably Efficient Reinforcement Learning with Linear Function Approximation*. 2019. arXiv: 1907.05388 [cs.LG]. URL: https://arxiv.org/abs/1907.05388.

[30] Mohammad Gheshlaghi Azar, Remi Munos, and Bert Kappen. *On the Sample Complexity of Reinforcement Learning with a Generative Model*. 2012. arXiv: 1206.6461 [cs.LG]. URL: https://arxiv.org/abs/1206.6461.

[31] Michael Kearns and Satinder Singh. "Near-optimal reinforcement learning in polynomial time". In: *Machine learning* 49 (2002), pp. 209–232.

[32] Lars Ankile et al. *From Imitation to Refinement – Residual RL for Precise Assembly*. 2024. arXiv: 2407.16677 [cs.RO]. URL: https://arxiv.org/abs/2407.16677.

[33] S. Garrido-Jurado et al. "Automatic generation and detection of highly reliable fiducial markers under occlusion". In: *Pattern Recognition* 47.6 (2014), pp. 2280–2292. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2014.01.005. URL: https://www.sciencedirect.com/science/article/pii/S0031320314000235.

[34] Minghuan Liu, Menghui Zhu, and Weinan Zhang. *Goal-Conditioned Reinforcement Learning: Problems and Solutions*. 2022. arXiv: 2201.08299 [cs.AI]. URL: https://arxiv.org/abs/2201.08299.

[35] Hadas Kress-Gazit et al. *Robot Learning as an Empirical Science: Best Practices for Policy Evaluation*. 2024. arXiv: 2409.09491 [cs.RO]. URL: https://arxiv.org/abs/2409.09491.

APPENDIX

A. Observations

TABLE VII: Observations used during training and their corresponding scaling factors. The same values were used across all the tasks. Every task uses a different set of observations.

Observations	***Scaling factor***
tcp_pos	100
tcp_orient	10
tcp_vel_pos	100
tcp_vel_orient	10
gripper_state	1
tcp_force	1
tcp_torque	10
tcp_pos_diff	100
joint_position	1
action	1
goal_box_position	10
box_position	10
box_orientation	10
goal_position	10

B. Lift

TABLE VIII: Observations used for the training of the lift task.

Observations	***Dimension***
tcp_pose	14
tcp_vel	12
gripper_state	4
tcp_force	6
tcp_torque	6
tcp_pos_diff	3
joint_position	12
action	14

TABLE IX: Rewards and costs used for the training of the lift task (SAC).

Reward	***Formula***
Action	$-0.1 \cdot \|\|a_t\|\|_2^2$
Speed	$-0.1 \cdot \|\|a_t - a_{t-1}\|\|_2^2$
Step	-0.1
Grasping	$1.5 \cdot \text{bool}(grasped)$
Suction	$-1.5 \cdot \text{bool}(not\_grasped)$
Orientation	$-25 \cdot \text{relu}(1.995 - \|\|\Delta\phi_1\|\|_2^2 - \|\|\Delta\phi_2\|\|_2^2)$
Position	$-10 \cdot \sum_i \lvert\Delta p_i - \text{sgn}(\Delta p_i) * 0.05\rvert$
Early end	-10
Success	$100 \cdot [\text{bool}(grasped) \wedge (p > p_{start} + 0.05)]$

TABLE X: Rewards and costs for the training of the lift task (RLPD).

Reward	***Formula***
Action	$-0.1 \cdot \|\|a_t\|\|_2^2$
Speed	$\|\|a_t - a_{t-1}\|\|_2^2$
Step	-0.1
Grasping	$1 \cdot \text{bool}(grasped)$
Suction	$1 \cdot \text{bool}(not\_grasped)$
Orientation	$-10 \cdot \text{relu}(1.995 - \|\|\Delta\phi_1\|\|_2^2 - \|\|\Delta\phi_2\|\|_2^2)$
Position	$-10 \cdot \sum_i \lvert\Delta p_i - \text{sgn}(\Delta p_i) * 0.05\rvert$
Distance	$-0.5 \cdot \frac{1}{\|\|p_{ee_1} - p_{ee_2}\|\|_2^2}$
Grasping	$15 \cdot \text{bool}(grasped) * p_z$
Early end	-150
Success	$200 \cdot [\text{bool}(grasped) \wedge p > p_{start} + 0.05]$

C. Reorientation

TABLE XI: Observations used for the training of the rotation task.

Observations	***Dimension***
tcp_pose	14
tcp_vel	12
gripper_state	4
tcp_force	6
tcp_torque	6
tcp_pos_diff	3
joint_position	12
action	14
box_orientation	3

TABLE XII: Rewards and costs used for the training of the rotation task (SAC).

Reward	***Formula***
Action	$-0.1 \cdot \|\|a_t\|\|_2^2$
Speed	$-0.1 \cdot \|\|a_t - a_{t-1}\|\|_2^2$
Step	-0.1
Grasping	$0.5 \cdot \text{bool}(grasped)$
Suction	$-0.5 \cdot \text{bool}(not\_grasped)$
Orientation	$-10 \cdot \text{relu}(1.995 - \|\|\Delta\phi_1\|\|_2^2 - \|\|\Delta\phi_2\|\|_2^2)$
Position	$-15 \cdot [\sum_i \text{bool}(grasped)\|\Delta p_i - 0.35\text{sgn}(\Delta p_i)\| + \sum_i \text{bool}(not\_grasped)\|\Delta p_i - 0.05\text{sgn}(\Delta p_i)\|]$
Rotation	$20 \cdot min(max(\phi_{box_t} - \phi_{box_{t-1}}, 0.015), 0.3)$
Early end	-10
Success	$200 \cdot [\text{bool}(grasped) \wedge (\|\Delta\Phi - 40°\|) < 5°]$

TABLE XIII: Rewards and costs used for the training of the rotation task (RLPD).

Reward	***Formula***
Action	$-0.1 \cdot \|\|a_t\|\|_2^2$
Speed	$-0.1 \cdot \|\|a_t - a_{t-1}\|\|_2^2$
Step	-0.1
Grasping	$0.5 \cdot \text{bool}(grasped)$
Suction	$-0.5 \cdot \text{bool}(not\_grasped)$
Orientation	$-30 \cdot \text{relu}(1.995 - \|\|\Delta\phi_1\|\|_2^2 - \|\|\Delta\phi_2\|\|_2^2)$
Position	$-15 \cdot [\sum_i \text{bool}(grasped)\|\Delta p_i - 0.35\text{sgn}(\Delta p_i)\| + \sum_i \text{bool}(not\_grasped)\|\Delta p_i - 0.05\text{sgn}(\Delta p_i)\|]$
Distance	$-0.1 \cdot \frac{1}{\|\|p_{ee_1} - p_{ee_2}\|\|_2^2}$
Rotation	$20 \cdot min(max(\phi_{box_t} - \phi_{box_{t-1}}, 0.015), 0.3)$
Early end	-150
Success	$200 \cdot [\text{bool}(grasped) \wedge (\|\Delta\Phi - 40°\|) < 5°]$

D. Lift+rotation

TABLE XIV: Observations used for the training of the lift & rotation task.

Observations	***Dimension***
tcp_pose	14
tcp_vel	12
gripper_state	4
tcp_pos_diff	3
joint_position	12
action	14
box_position	3
box_orientation	3

TABLE XV: Rewards and costs used for the training of the lift & rotation task (SAC).

Reward	***Formula***
Action	$-0.2 \cdot \|\|a_t\|\|_2^2$
Speed	$-0.2 \cdot \|\|a_t - a_{t-1}\|\|_2^2$
Step	-0.2
Grasping	$0.5 \cdot \text{bool}(grasped)$
Suction	$-0.5 \cdot \text{bool}(not\_grasped)$
Position	$-0.5 \cdot \text{bool}(p_t - p_0 < 0.05)$
Rotation	$40 \cdot min(max(\phi_{box_t} - \phi_{box_{t-1}}, 0.015), 0.3)$
Early end	-150
Success	$200 \cdot [\text{bool}(grasped) \wedge (\|\Delta\Phi - 40°\|) < 5°]$

TABLE XVI: Rewards and costs used for the training of the lift & rotation task (RLPD).

Reward	***Formula***
Action	$-0.2 \cdot \|\|a_t\|\|_2^2$
Speed	$-0.2 \cdot \|\|a_t - a_{t-1}\|\|_2^2$
Step	-0.2
Grasping	$0.5 \cdot \text{bool}(grasped)$
Suction	$-0.5 \cdot \text{bool}(not\_grasped)$
Position	$-0.5 \cdot \text{bool}(p_t - p_0 < 0.05)$
Rotation	$40 \cdot min(max(\phi_{box_t} - \phi_{box_{t-1}}, 0.015), 0.3)$
Distance	$-0.05 \cdot \frac{1}{\|\|p_{ee_1} - p_{ee_2}\|\|_2^2}$
Early end	-150
Success	$200 \cdot [\text{bool}(grasped) \wedge (\|\Delta\Phi - 40°\|) < 5°]$

E. Motion

TABLE XVII: Observations used for the training of the motion task. Policy (11) used the following set of observations.

Observations	***Dimension***
tcp_pose	14
tcp_vel	12
gripper_state	4
tcp_pos_diff	3
joint_position	12
action	14
goal_box_position	3
box_position	3
goal_position	3

TABLE XVIII: Rewards and costs used for the training of the motion task (SAC). Note that the *Goal distance* reward is computed as written in Eq. (3).

Reward	***Formula***
Action	$-0.1 \cdot \|\|a_t\|\|_2^2$
Speed	$-0.1 \cdot \|\|a_t - a_{t-1}\|\|_2^2$
Step	-0.1
Grasping	$0.3 \cdot \text{bool}(grasped)$
Suction	$-0.5 \cdot \text{bool}(not\_grasped)$
Orientation	$-4 \cdot \text{relu}(1.995 - \|\|\Delta\phi_1\|\|_2^2 - \|\|\Delta\phi_2\|\|_2^2)$
Position	$-4 \cdot \|\|goal\_box\_dist\|\|_2^2$
Goal distance	$60 \cdot norm(p_t^{box}, p_{t-1}^{box})$
Early end	-150
Success	$200 \cdot [\text{bool}(grasped) \wedge (\|\|goal\_box\_dist\|\|_2^2 < 0.05)]$

TABLE XIX: Observations used for the training of the motion task. Policy (12) used the following set of observations.

Observations	***Dimension***
tcp_pose	14
tcp_vel	12
gripper_state	4
action	14
goal_box_position	3
box_position	3
goal_position	3

TABLE XX: Rewards and costs used for the training of the motion task (SAC+RRL). Note that the *Goal distance* reward is computed as written in Eq. (3).

Reward	***Formula***
Action	$-0.3 \cdot \|\|a_t\|\|_2^2$
Speed	$-0.3 \cdot \|\|a_t - a_{t-1}\|\|_2^2$
Step	-0.2
Grasping	$2 \cdot \text{bool}(grasped)$
Suction	$-1 \cdot \text{bool}(not\_grasped)$
Orientation	$-10 \cdot \text{relu}(1.995 - \|\|\Delta\phi_1\|\|_2^2 - \|\|\Delta\phi_2\|\|_2^2)$
Position	$-10 \cdot \|\|goal\_box\_dist\|\|_2^2$
Goal distance	$80 \cdot norm(p_t^{box}, p_{t-1}^{box})$
Distance	$-0.5 \cdot \frac{1}{\|\|p_{ee_1} - p_{ee_2}\|\|_2^2}$
Early end	-150
Success	$200 \cdot [\text{bool}(grasped) \wedge (\|\|goal\_box\_dist\|\|_2^2 < 0.05)]$

F. Complete evaluation

TABLE XXI: Evaluation of the lift task. 30 episodes are performed for every policy.

Task	***Policy***	***Success Rate [%]***		***Grasp Sub-success [%]***		***Lift Sub-success [%]***		***Time [s]***	
		Seen	*Unseen*	*Seen*	*Unseen*	*Seen*	*Unseen*	*Seen*	*Unseen*
Lift	BT	77	**66**	**100**	99	77	66	**5.57** $\pm$ 2.54	**6.86** $\pm$ 2.20
	SAC	70	28	**100**	95	93	66	9.07 $\pm$ 5.14	11.79 $\pm$ 3.58
	RLPD	**100**	55	**100**	**100**	**100**	**95**	11.49 $\pm$ 3.00	12.88 $\pm$ 6.20

TABLE XXII: Evaluation of the rotation task. 30 episodes are performed for every policy.

Task	***Policy***	***Success Rate [%]***		***Grasp Sub-success [%]***		***Rotation Sub-success [%]***		***Time [s]***	
		Seen	*Unseen*	*Seen*	*Unseen*	*Seen*	*Unseen*	*Seen*	*Unseen*
Rotation	BT	60	62	**100**	98	60	62	7.80 ± 1.92	**6.95** ± 1.91
	SAC	**100**	75	**100**	**100**	**100**	**98**	8.94 ± 0.59	10.99 ± 3.74
	RLPD	**100**	**90**	**100**	**100**	**100**	95	**5.70** $\pm$ 0.38	8.52 ± 5.92

TABLE XXIII: Evaluation of the lift & rotation task. 30 episodes are performed for every policy.

Task	***Policy***	***Success Rate [%]***		***Grasp Sub-success [%]***		***Lift Sub-success [%]***		***Rot Sub-success [%]***		***Time [s]***	
		Seen	*Unseen*	*Seen*	*Unseen*	*Seen*	*Unseen*	*Seen*	*Unseen*	*Seen*	*Unseen*
Lift	BT	60	61	**100**	**97**	77	77	63	61	8.71 $\pm$ 1.31	**8.38** $\pm$ 1.66
&	SAC	**97**	28	**100**	**97**	97	55	97	50	7.71 $\pm$ 2.24	12.15 $\pm$ 3.86
Rotation	RLPD	**97**	**69**	**100**	95	**100**	**79**	**100**	**86**	**4.85** $\pm$ 2.25	8.49 $\pm$ 3.96

TABLE XXIV: Evaluation of the motion task. 30 episodes are performed for every policy.

Task	***Policy***	***Success Rate [%]***		***Grasp Sub-success [%]***		***Motion Sub-success [%]***		***Distance from goal [m]***		***Time [s]***	
		Seen	*Unseen*	*Seen*	*Unseen*	*Seen*	*Unseen*	*Seen*	*Unseen*	*Seen*	*Unseen*
Motion	BT	67	**55**	**100**	**100**	67	55	0.110	0.120	**8.80** $\pm$ 1.23	**9.06** $\pm$ 1.03
	SAC	40	30	**100**	98	53	47	0.072	0.089	11.62 $\pm$ 3.20	12.93 $\pm$ 4.25
	SAC + RRL	**70**	37	**100**	**100**	**80**	**56**	**0.063**	**0.083**	11.67 $\pm$ 4.61	14.19 $\pm$ 3.03