ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

RSL Robotic Systems Lab
Prof. Dr. Marco Hutter

Master Thesis

Towards Vision-Based Robotic Waste Picking Using Deep Reinforcement Learning

Spring Term 2023

Supervised by:
Hendrik Kolvenbach
Fidel Esquivel Estay
Mayank Mittal
René Zurbrügg

Author:
David Oort Alonso

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Towards Visual-Based Robotic Waste Picking Using Deep Reinforcement Learning

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

David	Oort Alonso

Student supervisor(s)

Hendrik	Kolbenbach
Fidel	Esquivel
Mayank	Mittal
René	Zurbrügg

Supervising lecturer

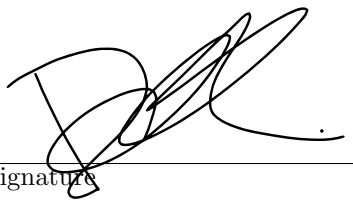
Marco	Hutter

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

David Oort Alonso, 16th May 2023

Place and date



Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Intellectual Property Agreement

The student acted under the supervision of Prof. Hutter and contributed to research of his group. Research results of students outside the scope of an employment contract with ETH Zurich belong to the students themselves. The results of the student within the present thesis shall be exploited by ETH Zurich, possibly together with results of other contributors in the same field. To facilitate and to enable a common exploitation of all combined research results, the student hereby assigns his rights to the research results to ETH Zurich. In exchange, the student shall be treated like an employee of ETH Zurich with respect to any income generated due to the research results.
This agreement regulates the rights to the created research results.

1. Intellectual Property Rights

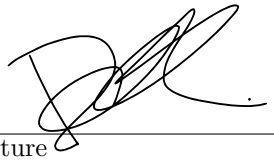
1. The student assigns his/her rights to the research results, including inventions and works protected by copyright, but not including his moral rights ("Urheberpersönlichkeitsrechte"), to ETH Zurich. Herewith, he cedes, in particular, all rights for commercial exploitations of research results to ETH Zurich. He is doing this voluntarily and with full awareness, in order to facilitate the commercial exploitation of the created Research Results. The student's moral rights ("Urheberpersönlichkeitsrechte") shall not be affected by this assignment.
2. In exchange, the student will be compensated by ETH Zurich in the case of income through the commercial exploitation of research results. Compensation will be made as if the student was an employee of ETH Zurich and according to the guidelines "Richtlinien für die wirtschaftliche Verwertung von Forschungsergebnissen der ETH Zürich".
3. The student agrees to keep all research results confidential. This obligation to confidentiality shall persist until he or she is informed by ETH Zurich that the intellectual property rights to the research results have been protected through patent applications or other adequate measures or that no protection is sought, but not longer than 12 months after the collaborator has signed this agreement.
4. If a patent application is filed for an invention based on the research results, the student will duly provide all necessary signatures. He/she also agrees to be available whenever his aid is necessary in the course of the patent application process, e.g. to respond to questions of patent examiners or the like.

2. Settlement of Disagreements

Should disagreements arise out between the parties, the parties will make an effort to settle them between them in good faith. In case of failure of these agreements, Swiss Law shall be applied and the Courts of Zurich shall have exclusive jurisdiction.

David Oort Alonso, 16th May 2023

Place and date



Signature

Contents

Preface

This thesis is part of the Autonomous River Cleanup Project, an initiative that aims to address the problem of waste streams in natural environments. The project has been running for three years, bringing together a diverse group of students, researchers, volunteers, and other individuals who share a common belief in the importance of protecting the planet.

I would like to take this opportunity to express my gratitude to my supervisors, Dr. Hendrik Kolvenbach, Fidel Esquivel Estay, Mayank Mittal and René Zurbrügg. Their support, encouragement, and guidance have been instrumental in the completion of this thesis. I would also like to thank Prof. Marco Hutter and the Robotic Systems Lab for providing me with an excellent research environment to carry out this work.

In completing this thesis, I extend my heartfelt appreciation to all the members of ARC, who have been exceptionally helpful and created a positive working atmosphere. Their contribution has been invaluable in the successful completion of this thesis. I am also grateful for the connections I have made through this project, the incredible spirit of collaboration, and the knowledge that has been shared with me.

In conclusion, I hope this thesis serves as a valuable contribution to the Autonomous River Cleanup Project and inspires others to join in the fight for a cleaner, healthier planet.

Abstract

The urgency for a circular economy underscores the need for advanced automation in waste management. Existing robotic systems, though effective in large-scale operations, struggle with diverse waste streams in pre-sorting stages.

In this thesis, we present an approach for robotic waste picking using deep reinforcement learning (DRL) with a focus on visual-based perception. Our primary goal is to develop a system that enables robots to efficiently and accurately sort waste in diverse and dynamic environments. We leverage NVIDIA's Isaac Gym, a high-performance physics simulator tailored for reinforcement learning tasks, to create a digital twin of the robotic system, which allows us to train complex grasping policies in a virtual environment.

Our methodology involves constructing a training pipeline and a virtual environment, where we progressively train the robot to tackle increasingly challenging grasping policies. These policies range from single-object static grasping to multi-object dynamic grasping with visual (RGB) end-to-end static policies. Our preliminary results show that an RL policy can grasp complex objects on a moving conveyor belt using object pose observations with a success rate of 87% and that the same architecture can learn to grasp a single static object using encoded RGB images of the environment with a 97% success rate and even generalize to unseen objects during training.

These promising results indicate that end-to-end visual-based reinforcement learning holds significant potential for addressing the waste sorting problem, paving the way for advanced robotic systems capable of efficiently managing heterogeneous waste materials in various scenarios, ultimately contributing to more sustainable and automated waste management solutions.

Symbols

Symbols

q	joint positions
$\dot{q}$	joint velocities
$\ddot{q}$	joint accelerations
π_θ	stochastic policy
s_t	state
a_t	action
R_t	reward
V_t	state value function
A_t	advantage function

Abbreviations

ARC	Autonomous River Cleanup
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DOF	Degrees of Freedom
DRL	Deep Reinforcement Learning
EE	End-Effector
GPU	Graphics Processing Unit
IK	Inverse Kinematics
MPC	Model Predictive Control
MSW	Municipal Solid Waste
OSC	Operational Space Control
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
SAC	Soft-actor-critic
SVM	Support Vector Machine
TRPO	Trust Region Policy Optimization
URDF	Unified Robot Description Format
COM	Center of Mass
FOV	Field of View

Chapter 1

Introduction

1.1 The Waste Management Problem

The global trend towards a circular economy and the mounting pressure towards more effective recycling (e.g. by the European Commission Waste Directive), calls for more automation of industrial waste management [1]. During the past two decades, we have witnessed the introduction of robots in industrial material recovery facilities, from optical sorters to suction-cup grippers (MaxAI[1], AMP Robotics[2], SamurAI[3], ZR Fast Picker[4]) and fingered grippers such as the ZenRobotics Heavy Picker [2]. These robotic units are very expensive and are only feasible in large centralized waste processing plants where they can handle large volumes of pre-sorted waste streams. However, there are still leaps of progress to be made for robots to reach the level of robustness and dexterity required to handle heterogeneous, piled waste at high speeds, in other words, the waste stream encountered at the presorting stage, where humans still play a major role. Such a system, if made low-cost, could enable decentralized sorting, which could for instance be deployed along rivers to address ocean-bound waste or in smaller waste management facilities in developing countries, where a majority of the sorting is still being done by humans. Workers in such facilities have to deal with highly unstructured and piled clutter and have to come up with collaborative strategies to sort through it. This is not only mind-numbing and physically repetitive work but is also dangerous and unhygienic.

1.2 Autonomous River Cleanup

The Autonomous River Cleanup (ARC) project is an ambitious initiative striving to address these challenges. By automating the waste management process, it promises a safer and more efficient solution for waste management facilities, especially in developing countries where human involvement remains significant. Automation not only shields workers from the hazardous and monotonous work conditions but also opens the door to increased operational efficiency and waste sorting purity.
Such improvements are crucial in pushing the boundaries of our current waste management practices. By increasing the value of unsorted waste, we can effectively reduce pollution in rivers and oceans. This not only leads to cleaner water bodies but also aids in the global transition towards a circular economy, where waste is seen as a resource rather than a burden.

[1] `https://www.max-ai.com/`
[2] `https://www.amprobotics.com/`
[3] `https://samur.ai/`
[4] `https://www.zenrobotics.com/fast-picker/`

This endeavor, however, is not without its challenges. The robotic systems at the heart of ARC need to operate in highly unstructured environments, dealing with a wide variety of waste materials. This brings up a plethora of problems related to perception, manipulation, and decision-making, further discussed below.

1.2.1 Challenges

The key challenges in object manipulation come from the fact that the environment not only needs to be geometrically and semantically understood and navigated through but there is a need for physical contact between the agent and the environment. This requires an implicit understanding of the physical properties of objects and the calculation of appropriate grasping points, collision-free grasps, and in some cases the discovery of grasps which are initially occluded or impossible (extrinsic grasping). Further, developing methods that scale to a wide range of tasks and for a wide range of objects, requires high robustness and many interactions in a simulated environment or the real world, both of which come with their own challenges. The challenges specific to our problem of robotic picking and sorting of waste are the following:

- Perception. The high clutter and diversity of the waste environment pose significant hurdles. Waste streams often contain objects that are entangled, mixed together, or piled up, obscuring a clear view of the individual items. This makes it difficult to fully understand the scene without physically interacting with it, such as shaking or spreading out the waste, or unwrapping items. Moreover, the diversity of waste materials adds another layer of complexity. The conveyor belt can present an endless variety of waste types, from plastics to metals, glass, paper, and organic materials, each with its own set of subcategories. Unlike in a controlled environment like a fulfillment center, each waste item can be transformed in arbitrary ways. Surface textures may exhibit oxidation, dirt, water, and other artifacts, and the geometric structure can be altered through tearing, squashing, folding, or breaking. This level of diversity and unpredictability requires advanced perception capabilities that can robustly handle such a high level of variation and uncertainty.

- Manipulation. On the other hand, becomes a daunting task in such a cluttered and diverse environment. Each category of waste comes with its own set of challenges, such as varying weights, sizes, shapes, and mechanical properties, which requires the robot to adopt a wide range of grasping and manipulation strategies. The robot must also be capable of performing pre- and re-grasp manipulations to disentangle items or to get a better grasp for subsequent picking actions. These manipulative interactions need to be fast and reliable, as they directly impact the efficiency of the waste sorting process.

- Moving targets. The objects to be interacted with and grasped are on a continuously moving conveyor belt, introducing challenges all the way from perception (motion blur, limited observations) to grasping (need to plan ahead and execute grasps in a time-critical manner for items not to leave the robot's workspace).

- Sim2real gap. Given the resources available, training a policy on the real system directly will not be feasible, meaning that it will be trained using interactions with a simulated environment and then tested on the real environment. This can easily fail if there are discrepancies between the real and simulated environment or if there is not sufficient randomization during training.

- Multi-agent collaboration. In a fully-deployed system, there will not be a single robot interacting with the objects on the belt but a whole chain of robot arms. Given enough computational resources, a centralized policy could be trained that is fed with observations from all robots simultaneously and outputs actions for all robots that maximize a reward for the system as a whole, which incentivizes collaborative behavior. However, depending on the number of robots used, this approach might become infeasible.
- Scalability & explainability. An end-to-end policy usually leads to robustness and performance benefits given that it bypasses any heuristics or assumptions that are not explicitly based on maximizing a return during training. However, it leads to challenges with respect to scalability and explainability. If raw sensor data is used as input to a policy, this usually means that this data is high-dimensional, and requires the training of policy networks with a large number of parameters, not to mention that it requires rendering many images in parallel. If a multi-agent policy is developed to encourage collaboration, it will require very heavy computing power or long training times.

1.2.2 The ARC Sorting Stage

The robotic sorting stage includes both custom hardware and software for the purpose of automated waste sorting and is the ultimate testbed where the policies trained in this work will be deployed at some point in the future. The current performance of the sorting stage is our benchmark and the inspiration upon which we build a digital twin of the environment.

The first version was setup in 2021 and successively improved since then and consists of three main components: a sensor for scanning, an apparatus for moving the waste stream along, and a robot for dynamically grasping and sorting the objects (see Figure 1.2).

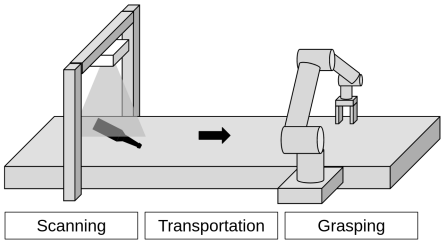


Figure 1.1: ARC sorting stage overview [3]

The current software stack is modular and linear meaning that it is split into different building blocks that sequentially transform an input into a processed output that gets passed to the block downstream. It employs various machine-learning tools for sensing, custom pick-and-place pipeline, and robot control, along with all the necessary communication nodes for the software and hardware components.

Object classification is the foundation of any sorting application. To identify waste objects based on RGB data, an accurate classification model is indispensable. The software stack incorporates two pipelines, as outlined in Table 1.1, to accomplish this task. MMdetection [4] is superior, having undergone additional training on a synthetic dataset explicitly produced for waste sorting, and is equipped with multiclass classification capabilities.

The movement of objects along the conveyor belt is monitored using the SORT algorithm [5] by tracking the RGB masks that function as input for the grasp pose estimation. GR-ConvNet [6] performs the grasp pose estimation by predicting the trash type and the planar grasp estimation comprising location, width, and angle. An RL-based task distributor [7] allocates the moving trash objects to each robot, optimizing the throughput. Finally, the pick-and-place node computes the rendezvous, grasp, and drop poses before communicating with the robot controller. An overview of the implemented frameworks is presented in Table 1.2.

Framework	**Model**	**Backbone**	**Dataset**	**Implement.**
Detectron2	Mask R-CNN [8]	ResNet-50 [9]	real	2021 [10]
MMDetection	Mask R-CNN [8]	Swin Transformer [11]	real synthetic	2022 [4]

Table 1.1: Object classification and instance segmentation implementations by ARC theses. [3]

Python	Python 3.8
ROS	Noetic
ML framework	Pytorch
Object detection	MMDetection / Detectron2
Object tracking	SORT
Grasp prediction	GR-ConvNet
Robot Control	MPC

Table 1.2: Software specifications and frameworks of latest version of the ARC Sorting Stage [3]

The sorting stage computer can get conveyor readings in real-time, allowing for feedback and adaptations during operation. Furthermore, a specialized self-calibration tool was developed in a previous thesis [3] to automate the calibration process between the detection camera and the robots.

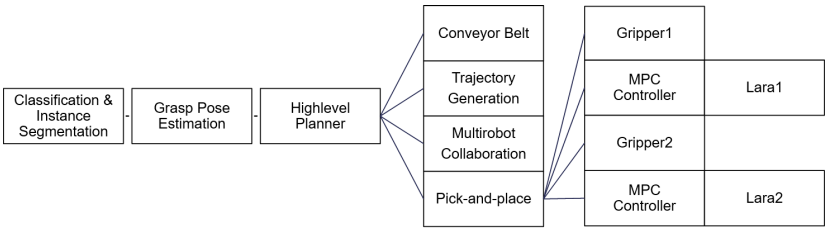


Figure 1.2: ARC sorting stage software components and communication overview [3]

The current approach at ARC has been tested on structured clutter environments (meaning that there are multiple objects on the belt, but they are not touching and are all on the same plane), at conveyor belt speeds of 5cm/s, reaching a Mean Picks Per Minute (MPPM) rate per robot of 30 with only top down grasps and in an open-loop manner, resulting in a grasping accuracy of 68%, sorting a single category. The use of a gripper for picking has led to the ability of this system to

grasp relatively complex shapes and surfaces (e.g. wet plastics). Human sorters at presorting stages of waste management facilities operate at around 40 MPPM on average in a significantly more complex environment (piled, wrapped trash with a range of weights and textures). This highlights the large gap to be bridged by more robust and performant autonomous sorting systems.

1.3 Related work

1.3.1 Previous work in robotic grasping

There is existing work done on dynamic grasping such as [12] that is reachability and motion-aware with motion prediction (which handles more than just linear trajectories). However, only 7 objects are tested in simulation and 3 of them in real-world tests. Moreover, the grasping on the real system is done on singulated objects moving in a linear motion of 4.46cm/s and using exclusively top-down grasps. The experimental setup in [12] includes a UR5 arm with Robotiq gripper along a single object conveyor belt.

Cleargrasp [13] fills in missing depth information for transparent objects by combining depth and RGB information from photorealistic simulation data and real data. In contrast to ClearGrasp, which uses photorealistic simulation that requires artists to create high-quality assets, SimNet [14] uses inexpensive RGB sensors and low-quality simulated data for training. This outperforms RGB-D sensors in the real world, especially on transparent or reflective objects. This has however only been shown with singulated and static objects.

kPAM 2.0 [15] leverages keypoint-based object representation for category-level pick-and-place and extends it to closed-loop manipulation policies with contact-rich tasks. However, this does not work on dynamic objects or tasks that require dense object geometry, which is bound to be the case in the vast object class that ends up in a waste conveyor belt.

1.3.2 Previous work on RL for robotic grasping

A significant area of focus in reinforcement learning research for robotic grasping is the development and utilization of visual representations. The use of visual inputs, such as images or video frames, allows the agent to extract rich, high-dimensional information from its environment, enhancing its ability to understand and interact with the world.
Nair et al. [16] utilized R3M, a visual representation pre-trained on diverse human video data, to facilitate data-efficient robotic manipulation learning. The approach led to substantial performance improvements over using standard image encoders such as Resnet [17].
Mosbach et al. [18] proposed a reinforcement learning framework for interactive grasping of various geometrically distinct objects. The approach demonstrated successful grasping even in challenging conditions, such as targeted grasping from a cluttered bin.
RL has also been used for extrinsic dexterity which involves using the external environment to solve a manipulation task that is initially impossible (occluded grasp) [19]. In [19], RL is used to train an agent that learns this emergent behavior by pushing a rigid object against a wall to be able to grasp it. However, this has only been shown with singulated and static objects and with a fixed wall placement. On top of that, the object 3D models were available for pose estimation, as well as the

target grasp on the object.

At ARC, RL has been used to solve for the optimal multi-robot picking strategy in terms of MPPM, deployed on the real system, and shown to improve the performance of the overall system over other picking strategies [7]. This work assumed that the objects were static on the moving conveyor belt and did not deal with the picking task which was done by a separate top-down grasping controller.

There has also been work that uses RL in an end-to-end fashion, by training visual policies. An example of this is QT-Opt [20], which is a scalable self-supervised vision-based RL framework that can leverage over 580k real-world grasp attempts to train a deep neural network Q-function to perform closed-loop, real-world grasping that generalizes to 96% top-down grasp success on unseen objects. Using only RGB vision-based perception from an over-the-shoulder camera, their method automatically learns re-grasping strategies, probes objects to find the most effective grasps, learns to reposition objects and perform other non-prehensile pre-grasp manipulations, and responds dynamically to disturbances and perturbations. A few details about the approach that might be key include: training an off-policy Q-function neural network with exclusively real-world data gathered over the course of 4 months, contrary to training an explicit actor, QT-Opt uses stochastic optimization over the critic to select actions and target values thereby avoiding issues with actor-critic instability. Effective off-policy training is valuable as it allows for rapid iteration on hyperparameters and architecture design without any data collection. However, additional on-policy joint finetuning consistently provides a quantifiable increase in performance with only about 28,000 additional grasps, reaching 96% grasp success. The grasping Q-function takes an RGB image as input, an action vector consisting of a cartesian vector, gripper rotation and open/close gripper commands, as well as a state vector with gripper aperture and gripper height. Since a completely random initial policy would produce a very low success with such an unconstrained action space, they use a weak scripted exploration policy to bootstrap data collection. This policy is randomized, but biased toward reasonable grasps, and achieves a success rate around 15-30%. They switched to using the learned QT-Opt policy once it reached a success rate of 50%. We are heavily inspired by this approach but choose to tackle it by training in a high-fidelity digital twin of the environment instead of in a parallelized real-world setup to which we do not have access to and is not as extensible. This could lead to similar emergent behaviors, but the additional challenges are that the objects would be dynamic and that there will be a sim2real gap to be bridged.

Another more recent example of a visual policy that required many fewer samples to train is C2F Q-attention [21]. This is the first time that a voxel-based representation is used for vision-based RL for 6D robot manipulation. Discretisation of rotation and gripper action is trivial given its bounded nature, but translation remains challenging given that it is usually a much larger space. They solve this problem via a coarse-to-fine Q-attention, where we start with a coarse voxelisation of the translation space, use 3D Q-attention to identify the next most interesting point, and gradually make the resolution higher at each point. Actions consist of a 6D (next-best) pose and gripper action, and the reward function is sparse, giving 100 on task completion, and 0 for all other transitions. The assumptions of this approach are: static targets, singular objects, well-defined tasks with demonstrations. It would be interesting to explore the extension of this work to dynamic and cluttered environments, and provide demonstrations with an easier example or with privileged information in simulation, or to use reward shaping (instead of sparse rewards) to learn without demonstrations and guide the exploration with continuous

rewards. The authors hypothesize that voxelizing image features instead of raw RGB data would perform better.
There has been a couple of extensions of C2F by the same authors, namely Learned Path Ranking (LPR) [22], which when given a next-best pose, learns to rank a set of paths generated from an array of path generating methods, including path planning, curve sampling, and a learned policy. The LPR is added as an extension to C2F-ARM, a highly efficient manipulation algorithm, and is able to achieve a wider set of tasks; in particular, tasks that require very specific motions that need to be inferred from both demonstrations and exploration data. The second extension is Q-attention with Tree Expansion (QTE) [23], which addresses the "coarse ambiguity" of C2F, which occurs when voxelization is significantly coarse, whereby it is not feasible to distinguish similar-looking objects without first inspecting at a finer resolution.

The above methods represent actions as next-best poses, but there has also been work on doing Joint Space control with Deep Reinforcement Learning [24], which shows that this method is capable of achieving similar error to traditional methods, while greatly simplifying the process by automatically handling redundancy, joint limits, and acceleration/deceleration profiles. Having a policy that directly maps to joint space commands, dispenses the use of a low-level controller relying on inverse kinematics/dynamics in simulation, which has the potential to introduce a sim2real gap in the case of a system training in sim and testing in real such as what we intend to do eventually.

1.4 Project Objective

Our intuition is that Deep Reinforcement Learning is well-suited for tackling the above-mentioned challenges, because while its very hard to map observations of a cluttered waste stream to robot commands, it is not hard to specify a reward function for the task of picking and sorting, and this is how RL allows to formulate the problem and through trial and error in simulation it can learn an arbitrarily complex mapping based on experience.
With the proliferation of photorealistic and highly parallelizable simulators such as Isaac Gym, we think it is way more scalable and extensive to build a digital twin of the environment rather than train directly on hardware. This also allows for hardware software co-design, and easily being able to evaluate the performance of different grippers against each other, sensor modalities, sensor placement, etc
The project goal of this work is defined as follows:

> "Take the first step towards end-to-end, visual-based RL for robotic picking"

Several sub-tasks are required to address all facets of the successful development of RL policies. The order of the presented sub-tasks corresponds to the planned timeline at which they are covered throughout the project.

1.4.1 Selecting a simulator

An essential first step in this project is selecting an appropriate simulator that can handle the complexity of the task and provide sufficient fidelity for training RL policies. The chosen simulator should support photorealistic rendering, be highly parallelizable, and efficiently simulate physics interactions, allowing the development of a digital twin of the environment for training and evaluation purposes.

1.4.2 Building a training pipeline and virtual environment

Once the simulator is selected, the next step is to create a training pipeline and virtual environment that closely replicates the real-world waste sorting setup. This includes designing the conveyor belt, importing suitable object datasets representative of a waste sorting environment, and positioning the robotic arm and sensors. The virtual environment should facilitate the iterative training and evaluation of various RL policies for robotic picking tasks through efficient parameter sweeping and model logging.

1.4.3 Training increasingly challenging grasping policies

The project's core objective is to train RL policies to tackle increasingly complex grasping tasks in the waste sorting domain. To achieve this, we plan to develop the following grasping policies:

Single-object, static grasping policy

We start by training a policy for grasping a single static object. This policy will provide a foundation for more complex tasks and serve as a baseline for performance comparisons.

Multi-object, static grasping policy

Next, we will train a policy capable of handling multiple types of static objects. This policy will need to generalize across a variety of different shapes and be able to grasp complex object such as deformed bottles that are common in real waste sorting scenarios.

Multi-object, dynamic grasping policy

The primary requirement of a policy that is deployed on the ARC sorting stage is that it needs to handle moving objects on a conveyor belt. This milestone involves trainiing such a policy which should adapt to the motion of objects and execute time-critical grasps effectively.

Visual (RGB) end-to-end static policy

Finally, we aim to develop an end-to-end visual policy that relies solely on RGB data. This policy is a stepping stone towards easily deploying an RL agent in the real system and have closed-loop feedback of the state of the conveyor belt in the robot workspace, which can lead to emergent behaviors such as pre-grasping and re-grasping.

1.4.4 Contributions

The contributions of this work can be summarized as follows:

- Selection and utilization of a suitable simulator for developing a digital twin of the robotic waste sorting environment.
- Development of a robust training pipeline and virtual environment for the RL agent to interact with, focusing on adaptability and scalability.

- Training and evaluation of various grasping policies, starting with single-object static grasping and moving towards multi-object dynamic grasping policies to handle the complexities of the waste sorting task.
- Implementation and evaluation of a visual (RGB) end-to-end static policy for robotic grasping, paving the way for further research in visual-based RL for robotic picking and sorting.

This report is structured as follows: Chapter 2 outlines the main methods used throughout this work including an introduction to Deep Reinforcement Learning and our simulation framework. Our results are presented and discussed in Chapter 3, which presents them in a sequential order, from single object static grasping to an end-to-end visual policy. Finally, Chapter 4 summarizes the main contributions and learnings from this work as well as outlining some potential avenues for future work.

Chapter 2

Preliminaries

There are multiple aspects to successfully enable vision-based grasping using deep reinforcement learning. We need an accurate digital twin of the sorting stage, an efficient simulation environment, and a capable reinforcement learning algorithm. In this chapter, we address each aspect of our approach, and set the stage for the results we obtained by training in this environment with the outlined algorithms.

2.1 Performance Assessment

In our end application of picking and sorting trash from a waste stream transported by a conveyor belt, we care about correctly-sorted MPPM. However, given that this is just the first step towards this end goal, we focus on maximizing a grasp success rate. In this work, we define a successful grasp as the ability to lift an object by a certain target height delta within a limited amount of time (usually 10-15 seconds). We make this choice because it is a simpler task to specify and learn and because once this task is achieved by an RL agent, we can simply use an IK controller to deposit the grasped object in the desired bin. The hard part is finding a grasp and locking the object in the gripper so that it can be lifted.

2.2 Deep Reinforcement Learning

Combining the principles of reinforcement learning with deep neural networks, deep reinforcement learning (DRL) trains agents to make informed decisions in complex and dynamic environments. The agent learns through feedback in the form of rewards or penalties, aiming to maximize the accumulated reward while interacting with the environment. DRL enables the agent to gain extensive knowledge about its environment and execute precise actions.

2.2.1 Robotic simulators for learning

The first step is to select a simulator on which to train agents. Table 2.1 summarizes the simulators considered and their pros and cons.

Isaac Gym and Isaac Sim

NVIDIA's Isaac Gym [25] is a high-performance physics simulator specifically designed for reinforcement learning tasks. Launched in 2021 as a free resource, it aims to boost sample efficiency and drastically reduce training times. NVIDIA

has achieved remarkable performance gains, enabling complex robotic tasks to be trained in mere minutes on a single GPU, as opposed to the hours or days required with other simulation frameworks. Although parallelization of physics engines is not a new concept, most architectures depend on a combination of CPU and GPU computations, necessitating substantial data transfers between them during training.
Isaac Gym addresses this bottleneck by providing fully GPU-accelerated training, allowing all training data to remain on the GPU and eliminating the need for transferring roll-outs of observation, reward, and action buffers between processing units. This massively parallel framework supports simultaneous roll-outs of thousands of environments, expanding the potential for reinforcement learning applications.
Building on the success of Isaac Gym, NVIDIA introduced Isaac Sim, a powerful and versatile robotics simulation platform that extends the capabilities of Isaac Gym. While Isaac Gym focuses on reinforcement learning tasks, Isaac Sim broadens the scope to encompass a wider range of applications, such as robotics development, testing, and deployment. Isaac Sim incorporates advanced features like high-fidelity rendering, realistic sensor models, and integration with NVIDIA's Omniverse platform for real-time collaboration and simulation. This comprehensive simulation platform enables researchers, engineers, and developers to create, test, and deploy cutting-edge robotic solutions in a virtual yet realistic environment.

Legged Gym

Concurrently with the launch of Isaac Gym, Legged Gym [26] was introduced as a learning platform for legged robots, offering all the necessary tools for rapid policy generation using Isaac Gym. Initially designed for locomotion tasks involving ANYmal [27], the repository has grown to include a variety of assets and pipelines. Efficient asset management and a user-friendly interface with Isaac Gym's Tensor API simplify the integration of new robots and environments. The repository also includes training buffer handling, loggers, and numerous training examples, making Legged Gym [26] an potential foundation for the robotic sorting environment.

ORBIT

ORBIT [28] is a unified and modular framework for robotics and robot learning, powered by NVIDIA Isaac Sim. It allows users to efficiently create realistic robotic environments with photo-realistic scenes and fast, accurate simulations. The framework offers a range of benchmark tasks with varying difficulty levels, supporting diverse observation and action spaces by including various fixed-arm and mobile manipulators, different controller implementations, and physics-based sensors.
Despite its potential, ORBIT was not chosen as the preferred simulator because it is relatively new and lacks the working examples and documentation available with more mature frameworks.

Isaac Gym Envs

Isaac Gym Environments (Isaac Gym Envs) is a collection of simulation environments built on top of NVIDIA's Isaac Gym, which was chosen as the foundation for our digital twin due to its numerous advantages. Isaac Gym Envs provides working examples of dexterous manipulation, making it a perfect fit for our application. Furthermore, the ability to easily modify robot morphologies allows for rapid experimentation and adaptation to different tasks. The platform also offers robust training infrastructure, including features for logging videos, conducting parameter sweeps, and more. These strengths make Isaac Gym Envs an ideal choice for developing a digital twin in the realm of robotics and reinforcement learning.

Simulator	Pros	Cons
Legged Gym	High-performance training of legged robots	Not mature for dexterous learning and visual learning
Orbit	High photorealism	Less working examples and documentation than more mature frameworks
IsaacGymEnvs	Working dexterous examples, ability to easily change robot morphologies, good training infrastructure for logging videos, sweeps, etc	Not highly photorealistic

Table 2.1: Pros and Cons of Considered Simulators. IsaacGymEnvs was the chosen simulator for this work.

2.2.2 An Overview of Proximal Policy Optimization

Introduced by OpenAI in 2017, Proximal Policy Optimization (PPO) [29] is an on-policy gradient method for reinforcement learning based on the actor-critic architecture. The actor-network learns to map states to actions, while the critic-network predicts the expected reward by learning a value function [30]. PPO's goal is to streamline the Trust Region Policy Optimization (TRPO) [31] approach while maintaining comparable performance. Although TRPO demonstrates better data efficiency than other policy gradient algorithms, solving higher-order optimization problems is computationally costly. To address this, PPO introduces a clipped surrogate objective that requires only first-order optimization, resulting in improved sample efficiency.
The main challenge of policy gradient methods is optimizing a policy to achieve maximum expected discounted reward while ensuring monotonic policy updates. Large steps during training can result in unstable behavior. TRPO addresses this issue by maximizing a surrogate objective, L^{CPI} (refer to Equation 2.1), with a constraint on the policy update size, or trust region (see Equation 2.2). This method effectively mitigates the problem of large policy updates but requires the computation of a higher-order problem (KL divergence). PPO, on the other hand, proposes clipping the probability ratio r_t to penalize ratios greater or smaller than one, yielding a new surrogate objective, L^{CLIP}. Taking the minimum of the clipped and unclipped probability ratio offers a lower bound on the unclipped objective, simplifying the optimization process. By doing so, PPO achieves a balance between constraining policy updates and maintaining computational efficiency, making it a popular choice for various reinforcement learning applications.
Let π_θ be the stochastic policy and $\hat{A}_t$ be an estimator of the advantage function at time-step t based on the value function.

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \right] = \hat{\mathbb{E}}_t[r_t(\theta)\hat{A}_t] \tag{2.1}$$

$$\hat{\mathbb{E}}_t[KL[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)]] \geq \delta \tag{2.2}$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{2.3}$$

In recent years, PPO has established itself as a stable and easy-to-implement learning algorithm that performs well across a wide range of tasks. For this work, we use a custom implementation of PPO from RL Games [32], a popular open-source

library of reinforcement learning algorithms. This implementation offers an optimal combination of performance and ease of use, making it suitable for our specific needs.

2.2.3 Specifications

The hardware and software specifications for this work are shown in Table 2.2.

Operating System	Ubuntu 20.04
Graphics	RTX 3080 and RTX 4090
Language	Python 3.8
Simulation	Nvidia IsaacGym Preview 4
Physics Engine	Nvidia PhysX
Gym Interface	Isaac Gym Envs
RL algorithm	PPO (rl_games)
ML framework	PyTorch 1.13 & Cuda 11.6

Table 2.2: Specification for DRL Training

2.3 Simulation

The simulation for grasping and sorting tasks is created in the Isaac Gym Envs framework described in section 2.2.1. It contains assets, buffers, and tools to enable high sample efficiency throughout the training. In the following sections, different approaches are discussed and the choice is reasoned.

2.3.1 Hardware Modelling

Figure 2.1: Neura Lara5 Robot. Hardware specifications can be found in appendix A

Figure 2.2: Robotiq 2f140 Gripper. Maximum opening: 140mm

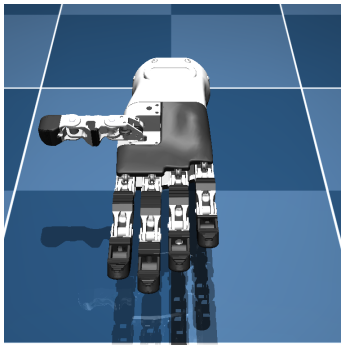


Figure 2.3: Schunk SIH Hand

Figure 2.4: Franka Panda parallel-jaw gripper. Maximum opening: 80mm

Robotic Arm

We import a URDF model of the Lara5 (see Figure 2.1), a 6-DOF robotic arm used in the ARC sorting stage. This URDF file contains an accurate kinematic model and matching inertia and point masses. Since currently we're training and testing in simulation, we can use any kind of controller in simulation to map from policy outputs to actual joint control inputs. However, if any of these policies were to be deployed to the real arm, matching the P-PI controller of the six Lara5 actuators would be required.

Isaac Gym Envs contains several other arms that can be swapped out through a configuration file, such as the Franka Panda arm.

Gripper

The ARC sorting stage uses the Robotiq 2f140 (140mm maximum opening) gripper on the Lara5 arm. The Robotiq 2f140 gripper (see Figure 2.2) is a parallel gripper consisting of two opposing parallelograms. The 6 moving links are controlled with a single position/velocity target. Isaac Gym prevents the use of close kinematic chains and URDF mimic joints are not supported, so mimicking the behavior of this gripper is not very easy. Hence, we decided to train on two off-the-shelf grippers in Isaac Gym Envs, namely the Franka Panda parallel-jaw gripper (see 2.4) and the Schunk SIH 5-finger hand (see 2.3).

2.3.2 Training Environment

There's two different environments for training that we've built inside Isaac Gym Envs. The first one is a static grasping environment, where the objects are placed on a table and the robot has to lift them to a target height (see fig. 2.5). The second one is a dynamic grasping environment, where the objects are placed on a conveyor belt and the robot has to also lift them to a target height (see fig. 2.6). The conveyor belt is moving at a constant speed, so the robot has to adapt to the changing position of the objects.

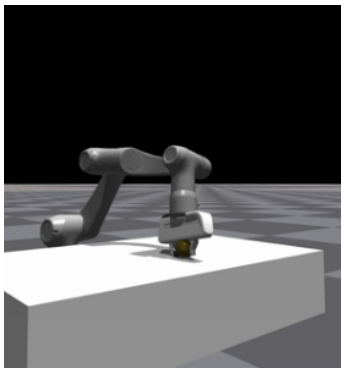


Figure 2.5: Simulation setup in the static grasping environment

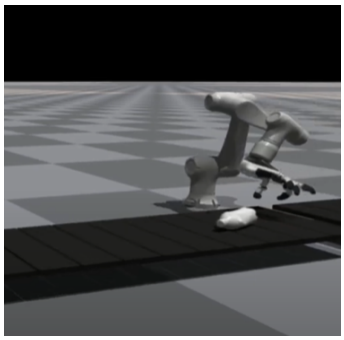


Figure 2.6: Simulation setup in the dynamic grasping environment

Agent

The agent is composed of the Lara5 with 6 DOFs, and either the Panda parallel-jaw gripper (2 DOF) or the Schunk SIH hand (5 DOF), counting a total of 7 DOFs that can be individually controlled. The gripper and arm combination can easily be swapped out for other arms and grippers in Isaac Gym Envs, as long as the URDF files are available. The training architecture automatically adapts to the number of DOFs of the agent.

Object Dataset

Acquiring the ability to grasp unfamiliar objects necessitates exposure to diverse object shapes and dimensions to facilitate domain transfer. The YCB dataset's lemon object represents the simplest training object (see fig. 2.7). Owing to recent advancements within ARC [4], a wide array of trash objects is available for random sampling. These objects pose considerable grasping challenges due to their irregular shapes and accurately represent the types of items encountered on the actual ARC sorting stage (see fig. 2.8).

Figure 2.7: Examples of objects from the YCB dataset (picture) [33]

Figure 2.8: Overview of the generated bottles dataset in a light and heavy deformation state. [4]

Object Representation

In order to effectively train the robotic system, various object representations are required. Isaac Gym Envs provides multiple object representations, including bounding boxes, object poses, and RGB images. Each representation offers a unique perspective on the environment and can be used to generate different grasping poli-

cies.
The lowest-dimensional representation is an object pose. This is cheap to obtain and allows for faster learning since the agent network is smaller, however it doesn't contain any semantic information about the object or information about its size. Bounding boxes provide a simplified representation of the object's spatial extent, making it easy to differentiate between objects with different shapes. RGB images provide a visual representation of the environment, capturing the appearance and texture of objects, which can be particularly useful for training end-to-end policies. The main advantage of RGB as a sensor modality is that this kind of observation is cheap and easy to get in the real world, which is not the case for object poses and bounding boxes. The downside of this representation is that it significantly increases training time for several reasons: images need to be rendered at each step, images have to be stored in memory which allows for significantly less parallelization of environments, the images have to be encoded either by a dedicated pre-trained encoder or by the agent itself, and lastly it is harder to learn a mapping between a high dimensional observation and control inputs than a lower dimensional observation.
Another possible option for object representation is point clouds. Point clouds are collections of data points in a 3D space, which can be generated from depth sensors or LiDAR. They provide a more detailed representation of the environment's geometry and can be useful for tasks requiring accurate perception of object shapes and structures. Isaac Gym Envs supports obtaining raw point clouds, synthetic point clouds (i.e. sampled points on the object surface) and semantic point clouds (adding an extra dimension to each point with an actor_id label).However, processing point clouds requires additional computational resources by encoding the raw list of points into a permutation-invariant embedding through for example PointNet++ [34]. Such representations, especially for simple grasping (versus for instance an in-hand reorientation task), only provides marginal gains [18], hence we did not explore this representation in this work.

Conveyor Belt

The implementation of a conveyor belt model is essential for transporting objects within the simulation. Instead of creating an overly complex model that includes cleats and rotating axes, a simplified version is employed to efficiently maintain a constant linear velocity for the objects throughout the episode roll-out.
To achieve this, the conveyor belt is modeled as an additional actor in the Isaac Gym simulation, where each panel on the belt is represented as a separate joint. These joints share the same configurable target speed, ensuring consistent object movement during the simulation. This approach strikes a balance between computational efficiency and accurate representation of the conveyor belt's functionality, making it suitable for training the robotic system in waste picking tasks.
See fig. 2.6 for a visual of the modelled conveyor belt.

2.3.3 Actions

In our reinforcement learning framework, the agent generates actions in the form of delta end-effector (EE) poses and delta joint angles for the gripper's degrees of freedom (DOFs). The delta EE poses are concatenated with the current EE pose, resulting in the desired EE pose for the next time step. These outputs are then processed by the Isaac Gym's built-in inverse kinematics (IK) controller, which employs the Damped Least Squares (DLS) IK method.
The DLS IK method is a widely used technique that computes the joint angle changes required to achieve the desired EE pose while maintaining stability and

avoiding singularities. By integrating the Isaac Gym's IK controller into the action pipeline, our RL agent can effectively generate and execute precise control commands for the robot's end-effector and gripper, allowing it to perform complex grasping tasks within the virtual environment.

2.3.4 Grasp Success

A well-defined grasp success criterion is essential for effective reinforcement learning. Instead of relying on complex criteria, such as requiring a specific gripper force, opposite force, object proximity, and gripper position, we adopt a more straightforward approach. We classify grasp success as lifting the object above a certain height, which simplifies the evaluation process and allows the agent to focus on grasping strategies.
To discourage undesirable behaviors like tossing the object, additional rewards that penalize the distance between the fingertips and the object are introduced. This method enables the agent to prioritize grasping before performing other tasks such as dropping, placing, or manipulating objects, leading to more efficient and effective training.
A detailed presentation of the rewards employed during training is presented in chapter 3 and summarized in table 3.1.

2.4 Training Process

In this section, we describe the training loop and simulation steps employed in the Isaac Gym Envs framework [25]. We utilized the default PPO hyperparameters in Isaac Gym Envs, as they provided satisfactory results and proved less critical than adjusting environment and reward parameters.

2.4.1 Simulation Steps

Simulation steps involve physics updates based on sampled actions and the calculation of advanced states and rewards. The process begins with sampling actions a_t from the policy π_t at state s_t. States comprise actor observations used for action prediction and privileged critic observations for estimating expected rewards. After processing the actions according to the control method (see 2.3.2), simulation physics are updated, and state tensors are refreshed. The policy frequency f_π runs slower than the simulation frequency f_s by a factor called control decimation CD. Following multiple physics steps, the simulation is rendered as needed, and state buffers are updated. Environments are continuously checked for termination conditions, such as reaching the maximum episode length or fulfilling task-specific conditions. Upon meeting a termination condition, the environment is reset to its initial state, and rewards r_t and advanced states s_{t+1} are calculated.

2.4.2 Curriculum

Curriculum learning is an effective approach to facilitate training in complex tasks by gradually increasing the difficulty of the task, thus allowing the model to build upon its acquired knowledge. This approach can be highly beneficial in various scenarios, as it enables better generalization, faster convergence, and improved robustness to initialization randomness. However, in our work, we decided not to employ curriculum learning. The primary reason behind this decision is that the task at hand has already been substantially simplified. Moreover, the randomness in the initial conditions was not severe enough to warrant the use of curriculum

learning.

In future work, incorporating curriculum learning might be a promising avenue to explore, especially if the task complexity increases or the initialization randomness becomes more pronounced. A well-designed curriculum could enable the learning process to progress more smoothly by gradually introducing new challenges, allowing the model to adapt and overcome them step by step. This strategy could lead to a more efficient and effective learning process and potentially yield better results in more challenging tasks.

2.4.3 Parameter Sweeps

In order to find the optimal training hyperparameters, such as target height, lift-off height, agent observations and the weights for the reward terms, we employed a parameter sweep methodology. Parameter sweeps involve systematically exploring a range of possible values for each hyperparameter and evaluating their impact on the learning process and the resulting policy performance. This approach allows us to identify the best combination of hyperparameters for the given grasping tasks leveraging computational performance.

Procedure

To conduct the parameter sweeps, we first defined a search space that includes the hyperparameters of interest. In our case, these were the target height, lift-off height, and the weights associated with various reward terms. For each hyperparameter, we specified a range of potential values to explore.
Next, we trained multiple instances of our reinforcement learning agent, each with a different combination of hyperparameter values. By evaluating the performance of these agents, we were able to identify trends and relationships between the hyperparameters and the overall success in the grasping tasks. Table 2.3 shows a summary of the hyperparameters sweeped, the distribution and ranges of the parameters and the fixed values that were chosen (if applicable) after initial experiments.

Parameter	Range	Distribution	Fixed Value
action_penalty	-5 to 2	log_uniform	0.0
fingertips_dist_penalty	-5 to 2	log_uniform	0.05
object_lift_off_reward	-5 to 2	log_uniform	0.25
success_bonus	4 to 9	log_uniform	1000.0
lift_off_height	0.025 to 0.1	uniform	–
target_height	0.075 to 0.25	uniform	–
learning_rate	-9 to -6	log_uniform	5e-4
max_episode_length	300 to 700	uniform	450

Table 2.3: Parameter sweep ranges, distribution types, and fixed values after initial experiments.

A visual representation of the parameter sweep can be seen in Figure 2.9, which illustrates the impact of various hyperparameter combinations on the learning process and grasping performance.
By fine-tuning these hyperparameters, we were able to achieve improved learning efficiency and robust performance in the grasping tasks. Our initial parameter sweeps allowed us to find more narrow ranges of values for the reward terms that were shared across tasks and made it faster to train well-performing policies.

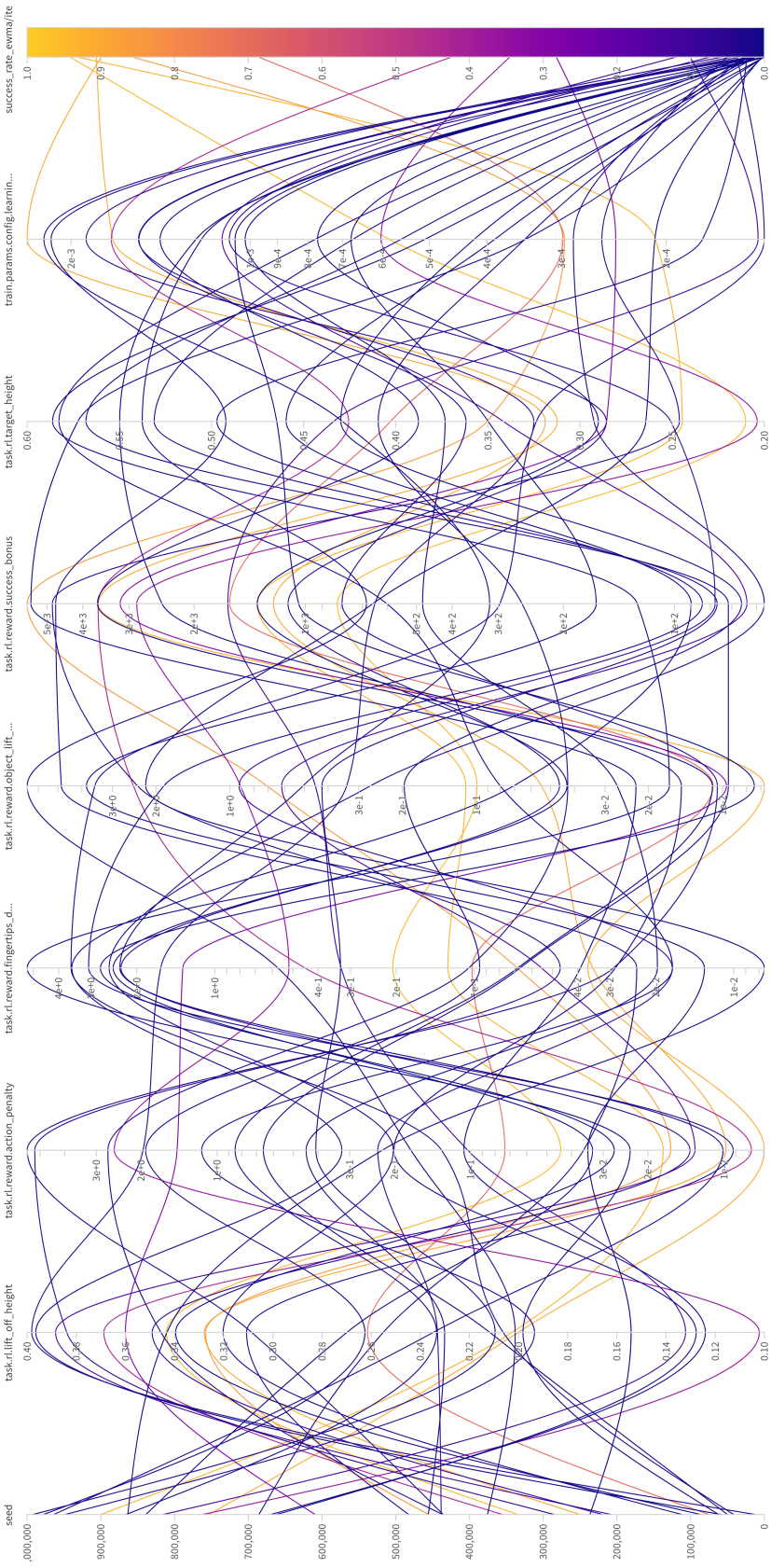


Figure 2.9: Example of a parameter sweep for hyperparameters, including target height, lift-off height, and reward term weights created using weights & biases. The color map indicates the success rate of the policy trained on a given combination of hyperparameters. This sweep is for the object lifting task on the ycb dataset with a bifinger.

Chapter 3

Method

The ultimate goal of our reinforcement learning research is the development of reliable policies that enable a 6-DOF robotic arm with a gripper to grasp unfamiliar, unstructured objects from a conveyor belt. This objective is multifaceted, involving a sequence of precise actions and various aspects such as robot control, object representation, grasp strategy, and reward structure. Confronting all these factors at once poses a formidable challenge.

As a result, we have decided to commence with a simplified task, primarily focusing on robot control. We then incrementally escalate the task difficulty, allowing for better understanding and tackling of each aspect. In this work, we introduce four tasks of progressively increasing complexity:

- **Grasping a single static object**: The initial task emphasizes fundamental robot control. The aim is to create a policy that successfully grasps a single stationary object, laying the groundwork for more advanced tasks. This stage also offers an opportunity to experiment with various object representations, hyperparameters and reward structures.
- **Grasping multiple static objects**: Building upon the previous task, this stage involves grasping multiple stationary objects. This added complexity enables the exploration of different grasp strategies and promotes the development of policies that can handle multiple object types and arrangements.
- **Grasping objects on a conveyor belt**: The most challenging task incorporates the conveyor belt, simulating a dynamic environment. Here, the robotic arm must adapt to the moving objects and successfully grasp them despite their continuous motion. This task necessitates further refinement of control strategies and grasp techniques to accommodate the added complexity and variability of the environment.
- **Visual end-to-end grasping**: The final task focuses on developing a visual end-to-end grasping policy. This approach aims to make the grasping process more robust by leveraging visual input to directly infer control actions. By incorporating visual data, the policy can better adapt to novel objects and environments, enabling a higher level of generalization and efficiency in real-world scenarios.

Robot morphology
We train these task on both the Franka parallel-jaw gripper and the Schunk SIH hand. Since these robot morphologies have different number of fingers, this affects the observation and action space as illustrated in fig. 3.2.

Initialization
By the start of the episode, the robot state is initialized at a default joint state. Additionally, the object pose is randomized by dropping the object from a fixed height and a standard deviation of 5cm in either lateral direction (so a box of 5cm in standard deviation).

Termination
An episode is terminated if the maximum number of steps is reached (350) or when the object reaches the target height. No explicit termination happens when the object falls off the table. A penalty for this could encourage the agent to keep the object in the workspace and could be a sensible addition.

Reward
All of the tasks explored in this thesis share the same reward structure. We employ a combination of rewards to guide the agent's behavior during training. The various rewards are designed to encourage specific aspects of grasping performance:

1. *Action penalty*: This is a dense reward that penalizes the agent based on the norm of the action. It helps to promote smooth and efficient control of the robot by discouraging large or abrupt actions.

2. *Fingertips distance penalty*: This dense reward is inversely proportional to the distance between the fingertips and the target object COM, with a scaling factor and an offset constant. It encourages the agent to move the gripper closer to the object, which is essential for successful grasping.

3. *Lift-off reward*: Another dense reward, it is inversely proportional to the height of the lifted object above a certain threshold, with a scaling factor and an offset constant. This reward motivates the agent to lift the object off the ground, which is a critical aspect of successful grasping and manipulation.

4. *Success bonus*: This is a sparse reward provided as a constant bonus when the agent successfully completes the grasping task. It serves as a strong signal to reinforce the desired behavior of grasping and lifting the object.

By combining these rewards, we ensure that the agent receives adequate guidance throughout the learning process, balancing dense rewards that provide informative gradients and sparse rewards that signal task completion. This combination of rewards aims to improve the efficiency of learning and the overall performance of the trained policy.

Reward	**Formula**
Action penalty	$\|.\|$
Fingertips dist penalty	$scale * \frac{1}{(dist+c_1)}$
Lift-off reward	$scale * \frac{1}{(height+c_2)}$
Success bonus	const

Table 3.1: Rewards for both moving and static grasping. *dist* is the average distance between each fingertip of the gripper and the object COM and *height* is the difference between the starting height of the object COM and the current height in a fixed world frame. $c_1 = 0.1$ and $c_2 = 0.02$.

3.1 Grasping Singulated Static Objects

The simplest task we could come up with for the agent is to grasp a static object and lift it by a certain target height (around 15cm) within a roll-out length of $7s$, which corresponds to 350 simulation steps.

Observation

For the single object grasping task, we use object poses as part of the observation to provide the agent with crucial information about the target object's position and orientation. This enables the agent to make informed decisions while trying to grasp the object. Including object poses in the observation is particularly important for this simplified task, as it allows the agent to focus on learning the grasping mechanics and understanding the relationship between the object and the robot.

In addition to the object poses, we also concatenate proprioceptive observations of the robot state, which include the end-effector pose and the fingertips pose. By incorporating these features into the observation, the agent can develop a better understanding of the robot's current configuration and how it relates to the target object.

The exact dimensions of the observation vector are displayed in fig. 3.2. An overview of the learning architecture for this task including the Actor and Critic network architecture is shown in fig. 3.2.

3.1.1 Results & Discussion

Training on a single object

We begin by presenting the results of training on the lemon object from the YCB dataset (refer to fig. 2.7). As shown in fig. 3.3, the success rate achieved for both the parallel-jaw gripper and the Schunk SIH hand is quite high (98% and 97% respectively). This demonstrates the potential of our single framework to train policies for different gripper morphologies. Furthermore, these results suggest that using only object poses as observations is sufficient to achieve a high success rate for grasping tasks, even though overfitting to a single object shape may occur.

The training plots in fig. 3.3 reveal that a variety of combinations of training parameters and reward weights can lead to high success rates for tasks like this one. This indicates that our method is robust to different hyperparameter choices and is capable of adapting to various configurations.

Training on multiple objects

Next we increase the complexity of the task by training on the full YCB dataset, see fig. 2.7 for a few example objects from the dataset.

It is noteworthy that utilizing bounding boxes (bboxes) did not yield improved performance over object poses in this task, as shown in fig. 3.4. One possible explanation for this observation is that learning from a lower-dimensional pose is more manageable, especially since object size can be inferred from a combination of object pose and fingertips pose. Additionally, it is conceivable that the standard orientation of the object axes, whereby the long side consistently aligns with the z-axis, could have allowed the agent to implicitly learn more about the object shape and the best grasp orientation from just a pose. The agent's grasp is adept, as evidenced by its persistent attempts to grasp and move the object, thereby facilitating the identification of an optimal grasp configuration.

The YCB objects are a nice test to get a sense of the robustness of the RL policy since it contains a high diversity of object shapes, however, most of these objects are not representative of what would be encountered on the real ARC sorting stage which mostly processes waste such as bottles. Hence, our next step is to train on

the synthetic ARC trash dataset which contains deformed objects such as bottles, see fig. 2.8.

The objects in this experiment were more challenging and diverse due to the usage of the most challenging variant of the ARC bottles dataset, containing highly deformed bottles. As shown in fig. 3.5, the parallel-jaw gripper had a significantly lower success rate, averaging at 25%, in comparison to the five-finger gripper, which had an average success rate of 80%. This discrepancy may be attributed to the inherent redundancy of the five-finger gripper, which allows for a greater number of contact points to grasp objects with irregular shapes. The results demonstrate the power of the RL training framework, which not only trains grasping policies but also facilitates informed hardware decisions based on the nature of the dataset. It is worth noting that each line on the plot corresponds to a different training run with distinct hyperparameters. Overall, the results suggest that the five-finger gripper is more suitable for grasping highly deformed bottles than the parallel-jaw gripper.

3.2 Grasping Moving Objects

To investigate the effectiveness of dynamic grasping policies, we conducted an experiment where we placed the deformed bottles arc dataset on a conveyor belt moving at 5cm/s and grasped it with the 5-finger gripper. Our aim was to evaluate the agent's ability to grasp moving objects. The results we obtained were promising, with a success rate of 87% achieved for a target lift height of 10cm. The grasp strategy that the agent learned was a bifinger picking strategy, which is interesting as it differs from the grasp strategy learned by the static grasp policy.

It is worth noting that the success rate achieved in this experiment is higher than that of the static grasp policy, which is likely due to the lower target height at which it was trained and the fact that episodes were longer, giving the agent more opportunities to lift the object.

Another advantage of the dynamic grasping policy is that we did not need to use a pose or action history to train it. We used the same observations, actions, and rewards as in the previous experiment, which meant that the agent could implicitly learn the conveyor belt speed. This again shows the extensibility of the RL training framework developed.

Overall, our results suggest that the agent is capable of grasping moving objects with a high success rate which is a promising first step towards deploying such a policy to the real ARC sorting stage. The results are presented in fig. 3.7, which shows the success rate as a function of training time for dynamic grasping policies trained on the ARC deformed bottles dataset with the five-finger Schunk gripper. Additionally, fig. 3.6 shows a typical grasping sequence for the learned policy on a sample bottle.

3.3 Grasping a Static Object Visually

While operating on state-space observations has been effective for training robotic agents, it presents certain limitations when attempting to achieve human-level performance in end-to-end robotic waste picking. This is primarily due to the fact that state-space observations are difficult to obtain in real-world, cluttered environments and lack the richness of visual sensor data. Therefore, we believe that transitioning to visual observations is a necessary step to address these challenges and enhance the capabilities of our robotic agents.

3.3.1 Motivation for Switching to Visual Observations

Visual observations offer several advantages over state-space observations. First, they provide rich sensor data, such as RGB images, which contain information about the texture, speed, and shape of objects. This information can be crucial for inferring the weight, consistency, and surface friction characteristics of objects, as well as determining their motion through motion blur. This level of detail is simply unattainable through state-space observations.
Additionally, visual observations allow for truly end-to-end RL agents, which do not require an external system to estimate object poses. This makes the robotic agents more versatile and adaptable, particularly in real-world scenarios.

3.3.2 Challenges and Architectural Changes

Incorporating visual observations into our RL policy brings its own set of challenges. The high-dimensional input must be mapped to high-dimensional control outputs on a robot arm, requiring more complex processing and understanding. To address this challenge, we used the r3m encoder [16]. This off-the-shelf encoder is based on a ResNet-18 architecture and pre-trained on the Ego4D dataset. This encoder demonstrated significantly better performance on a behavior-cloning manipulation task due to the inclusion of real-world data of humans manipulating objects.
The architectural changes were focused on the observation side, as illustrated in fig. 3.9. A virtual RGB camera with a FOV of 87 deg was placed was placed in front of the robot workspace and captured single frames at each step of the simulation. This increased simulator step times (by a factor of around 10x) as images had to be stored in memory, reducing the maximum number of parallel environments from 16k to 512. Additionally, encoding the images added an extra computational step.

3.3.3 Results and Generalization

Despite the challenges, the trained policy showed satisfactory results and even demonstrated the ability to generalize to unseen objects from the YCB dataset (see table 3.2). As can be seen in fig. 3.10, learning from RGB images proved to be significantly more difficult due to the higher dimensional input. Training times for a successful grasp were about 150x longer than for the state-space policy (from 1hr to 6 days). This however is for one set of hyperparameters as we did not have time to run a sweep for the visual-learning task. The reason that this could have taken so much longer is first of all that the step time in the simulation is already a factor of 10x longer, secondly, due to memory limitations of our hardware (RTX 4090) it wasn't possible to do the same amount of parallelization (or experience gathering) per step (512 parallel environments vs 16k in the state-space case). Finally, learning from a higher-dimensional unstructured vector such as an image embedding is a lot harder than a lower dimensional-state space representation, so it makes sense that the agent took longer to learn the relationship between that embedding and the rewards. It's also important to remember that r3m was not pre-trained on images from our simulator, so this could potentially accelerate learning in the future.
The generalization ability of the policy to unseen objects holds for objects of similar shape to the training object (lemon) but different aspect/texture. The policy has overfitted to some extent to the round shape of the lemon and is unable to grasp the apple or the pear. The policy is also sensitive to camera position changes (see table 3.3), which suggests that future work could involve training on noisy camera positions to improve robustness against camera miscalibrations.
In conclusion, our transition to visual observations has made significant progress towards achieving human-level end-to-end robotic waste picking. The use of visual

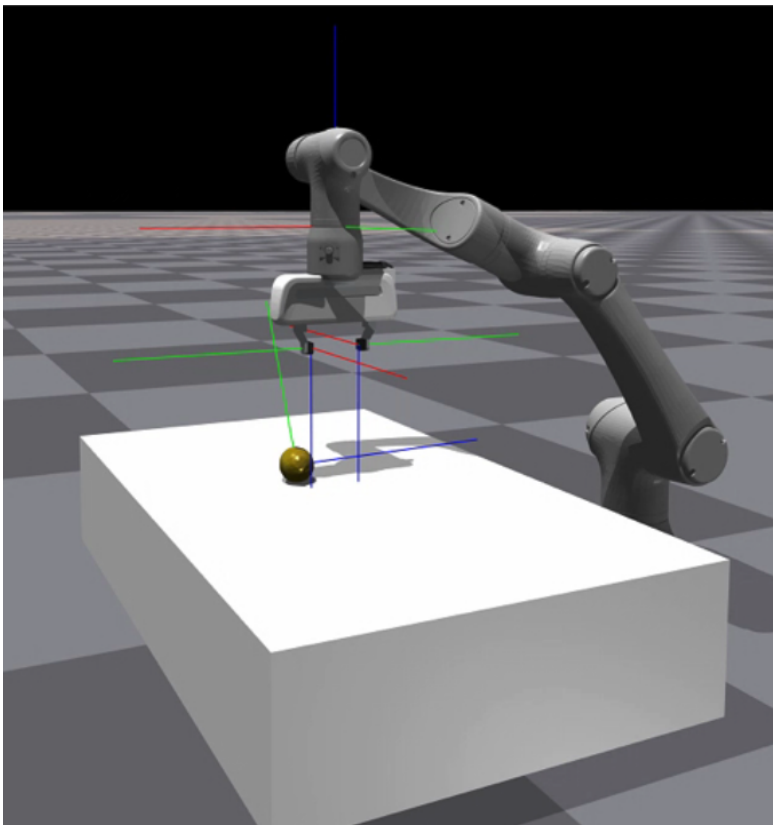
inputs addresses the limitations of state-space observations and provides richer data for the robotic agents to operate on, ultimately leading to improved performance in real-world applications.

Object	**Success Rate**
Lemon	0.97
Peach	0.92
Plum	0.99
Apple	0
Pear	0

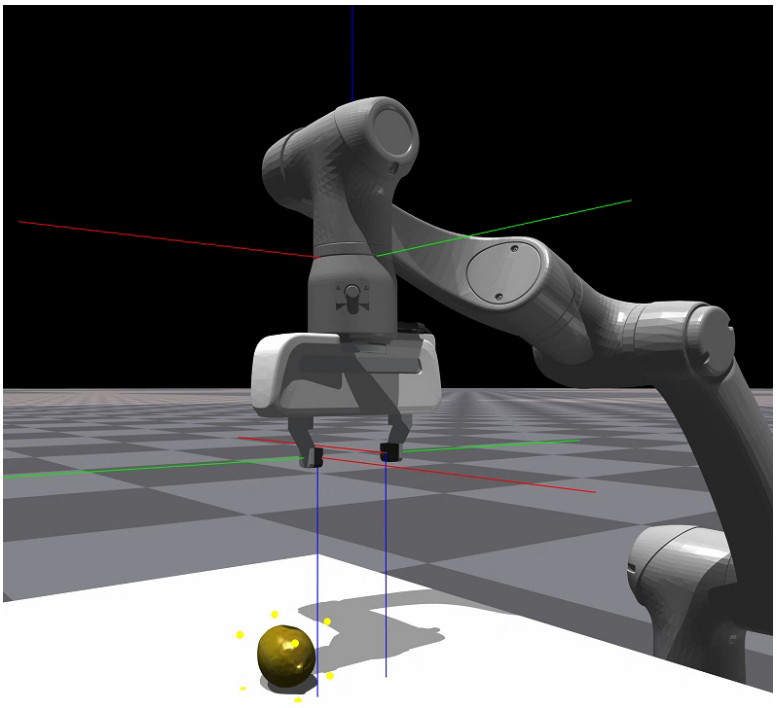
Table 3.2: Evaluation and generalization performance of the trained visual policy on static singulated objects from the YCB dataset. Note that the agent has only seen the Lemon object during training.

Object	**Success Rate**
Lemon (training object)	0.05
Peach	0.1
Plum	0.25
Apple	0
Pear	0

Table 3.3: Miscalibrated camera evaluation and generalization performance of the trained visual policy on static singulated objects from the YCB dataset. The camera is miscalibrated by 2mm longitudinally and 1mm laterally. Note that the agent has only seen the Lemon object during training.



((a)) Pose Representation. Each axis represents a pose observation. A quaternion (seven-dimensional representation) is used.



((b)) Bounding box representation. Corners are shown as yellow dots. The 3D position of the eight points is concatenated into a single vector.

Figure 3.1: Illustration of the state-space observations used. Refer to fig. 3.2 for a full list of observations and their dimensions.

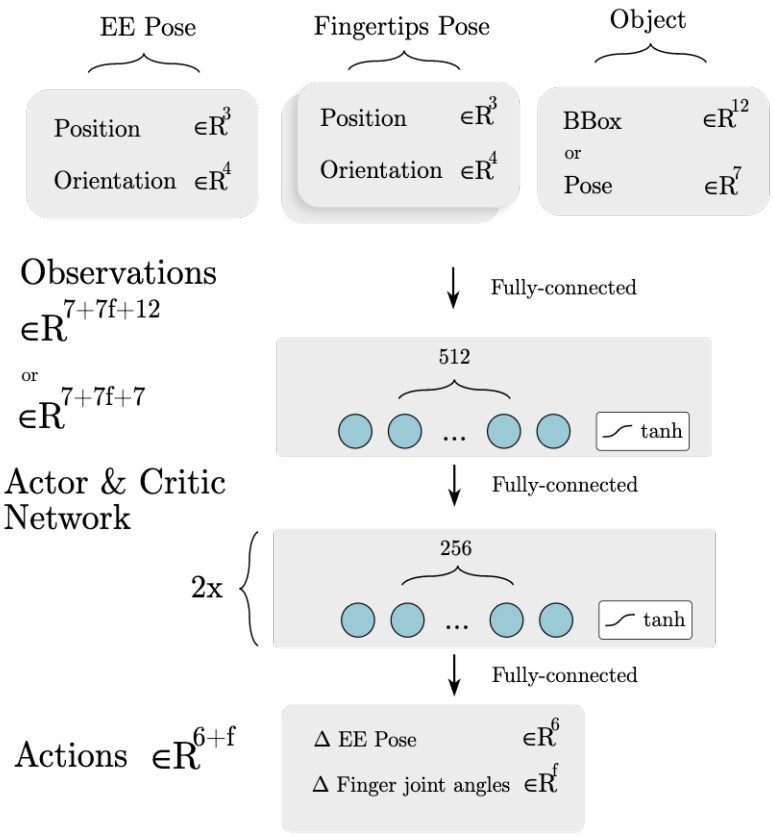
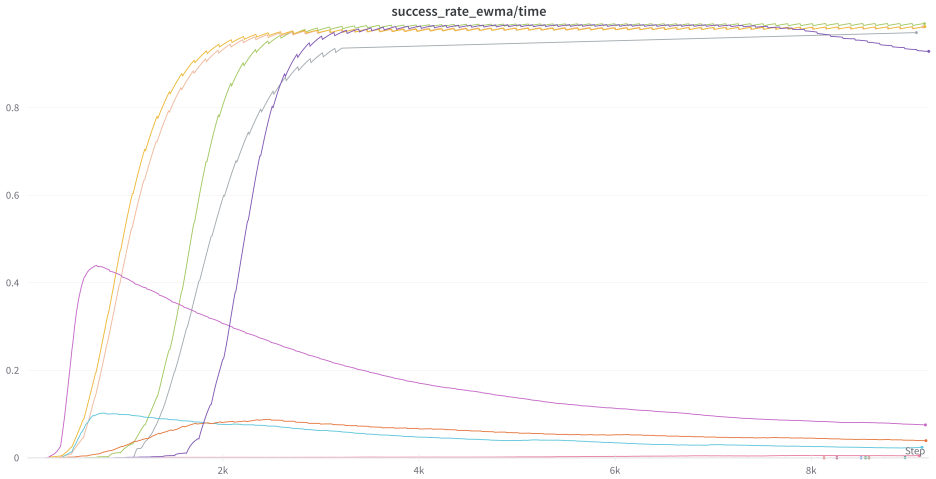
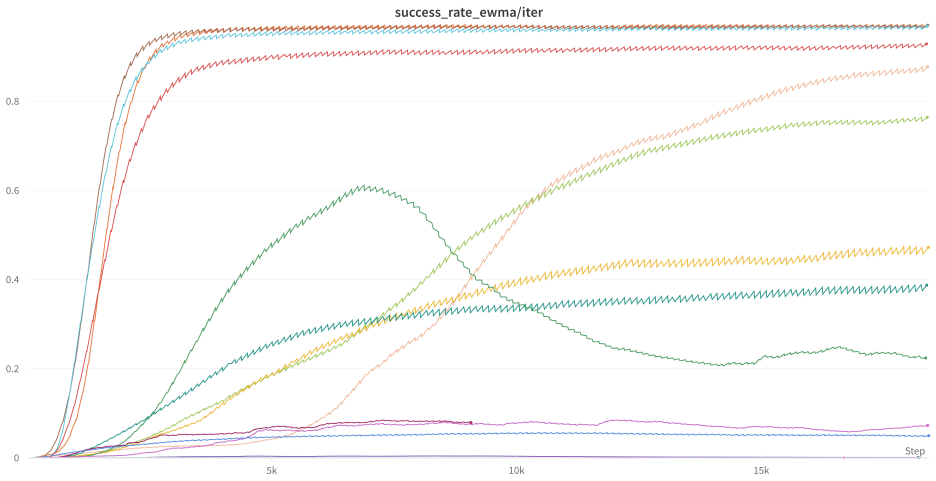


Figure 3.2: Overview of the learning architecture for state-space observation tasks. This includes all static grasping and dynamic grasping tasks. f is the number of fingers of the gripper (e.g. 2 for the parallel-jaw gripper and 5 for the Shunk hand).

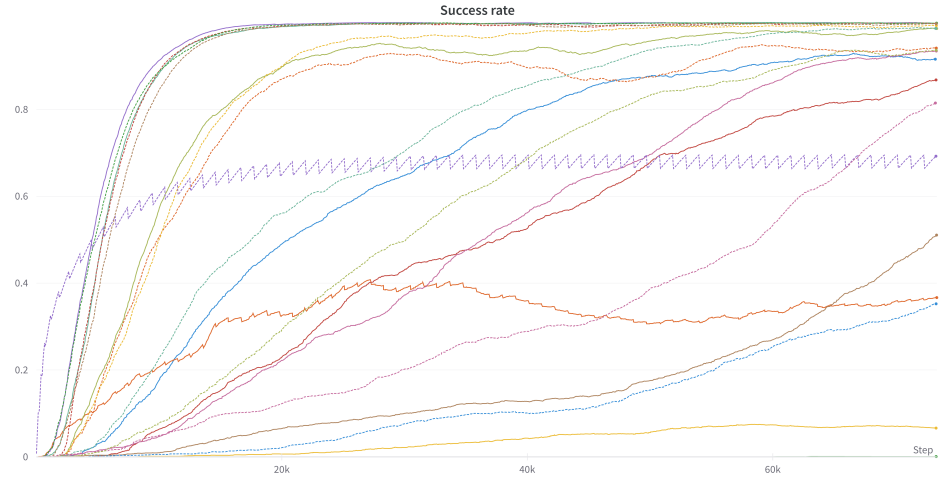


((a)) Parallel-jaw gripper (98% success rate)

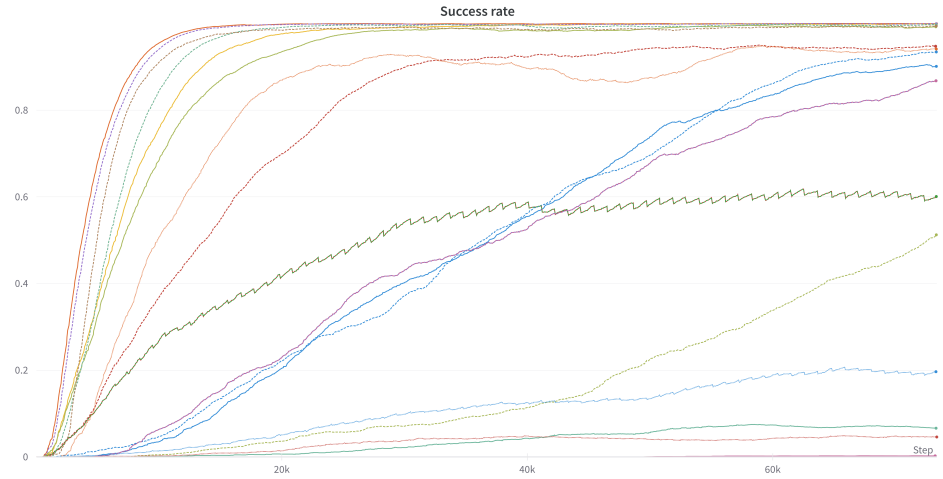


((b)) Schunk SIH hand (97% success rate)

Figure 3.3: Success rate as a function of training time for a single static lemon grasping policy with different grippers. Each line represents a training run with different hyperparameters. See table 2.3 for a summary of the hyperparameters sweeped and their respective ranges.

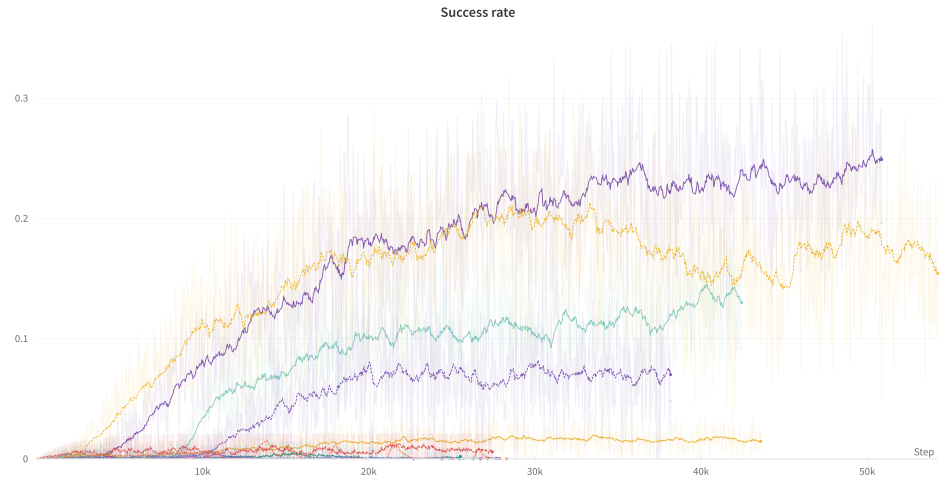


((a)) Pose representation. Average success rate of 56%

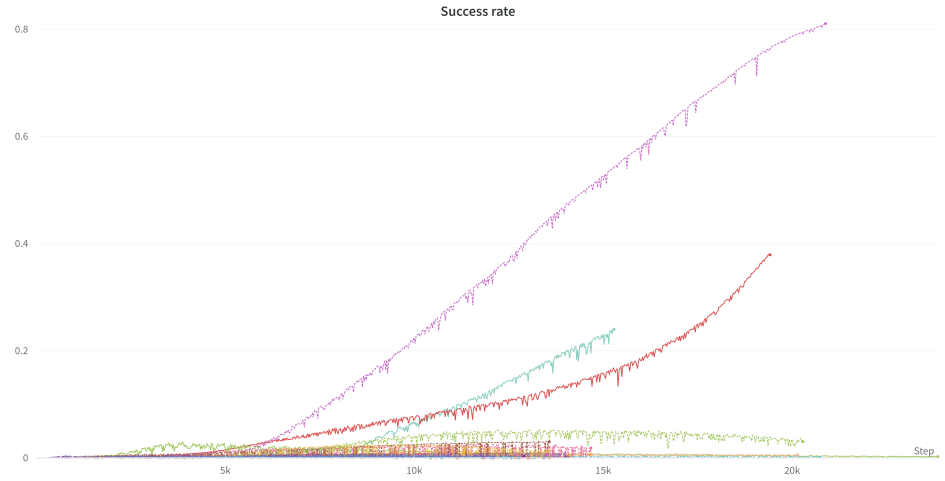


((b)) Bounding box representation. Average success rate of 60%

Figure 3.4: Success rate during training for different objects from the YCB dataset with the parallel-jaw gripper in a static environment (each line is a different object). The objects that perform best are rounded objects such as the fruit items, and those that are ungraspable are either too large to fit in the gripper (e.g. cereal box in fig. 2.7) or have a shape out of distribution (e.g. scissors). As can be seen, some objects are grasped with close to 100% success rate while others are ungraspable. There's no significant difference between the average success rate of the pose representation and the bounding box representation



((a)) Pose representation with the parallel-jaw gripper. Highest average success rate: 25%



((b)) Pose representation with the five-finger gripper. Highest average success rate: 80%

Figure 3.5: Success rate as a function of training time for policies trained on the ARC deformed bottles dataset with different grippers. Each line represents a training run with different hyperparameters. It can be noted that some runs are still increasing in performance when cut off at the maximum number of iterations. Some runs are inadvertedly cut off because training crashes due to memory limits, or other issues. Continuing to train these runs from a checkpoint would be a possibility to further enhance performance but is beyond the scope of this work. The highest performing hyperparameters are summarized in fig. 2.9, those which are not fixed tend to perform better as the task gets easier (e.g. lower target height).

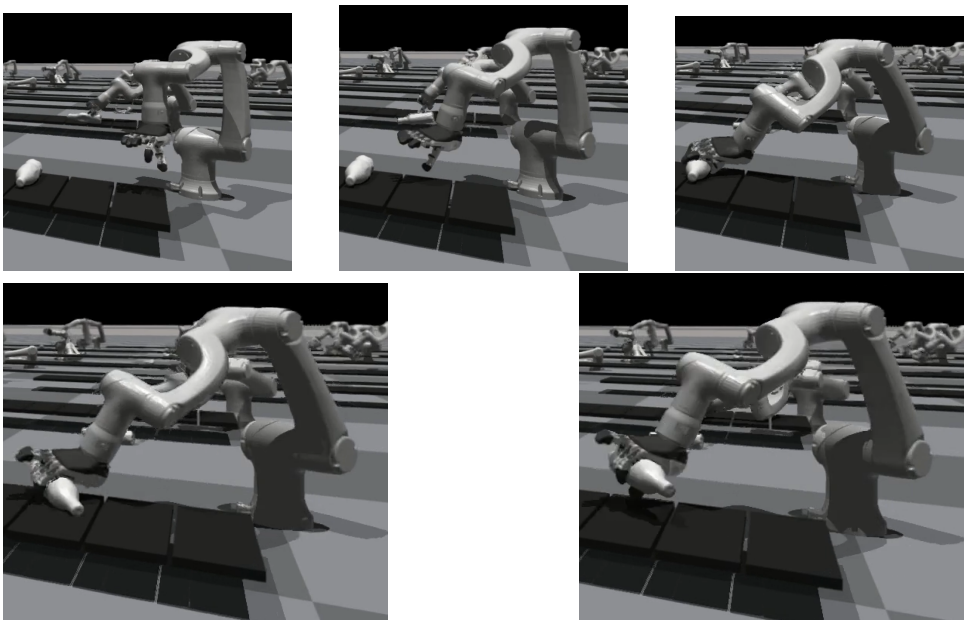


Figure 3.6: Grasping sequence for an example object from the ARC dataset placed on a moving conveyor belt. The two finger pinching strategy that emerges can be clearly seen in the five-finger gripper.

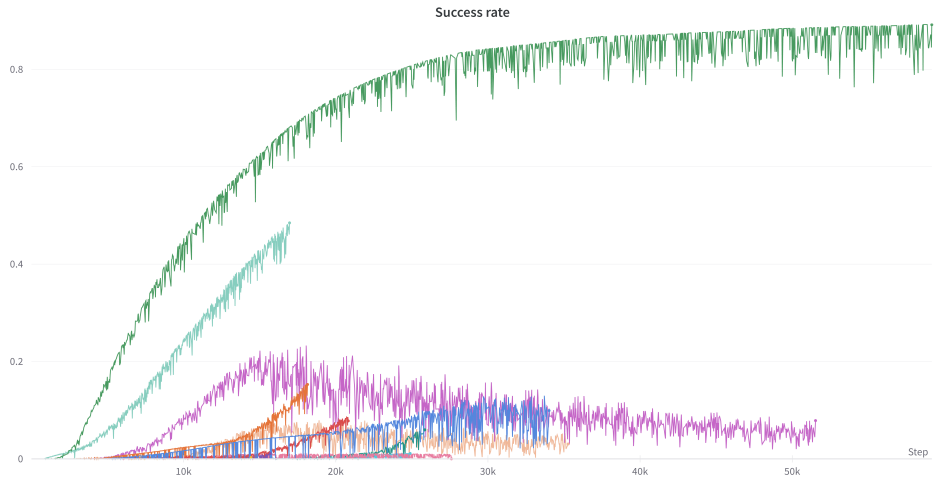


Figure 3.7: Success rate as a function of training time for dynamic grasping policies trained on the ARC deformed bottles dataset with the five-finger Schunk gripper. Each line represents a training run with different hyperparameters.

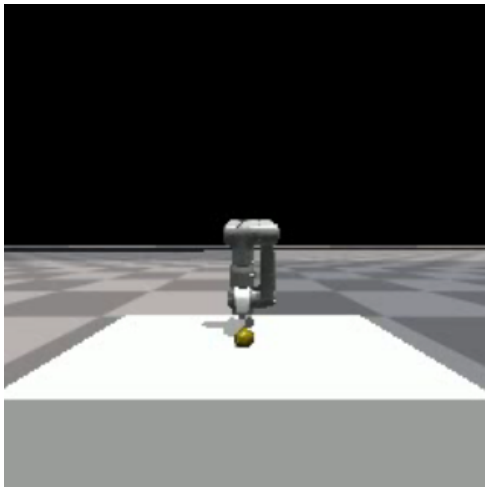


Figure 3.8: Sample 224x224 RGB image passed to the visual agent as observation. The camera is placed in front of the robot workspace and has a FOV of 87 deg.

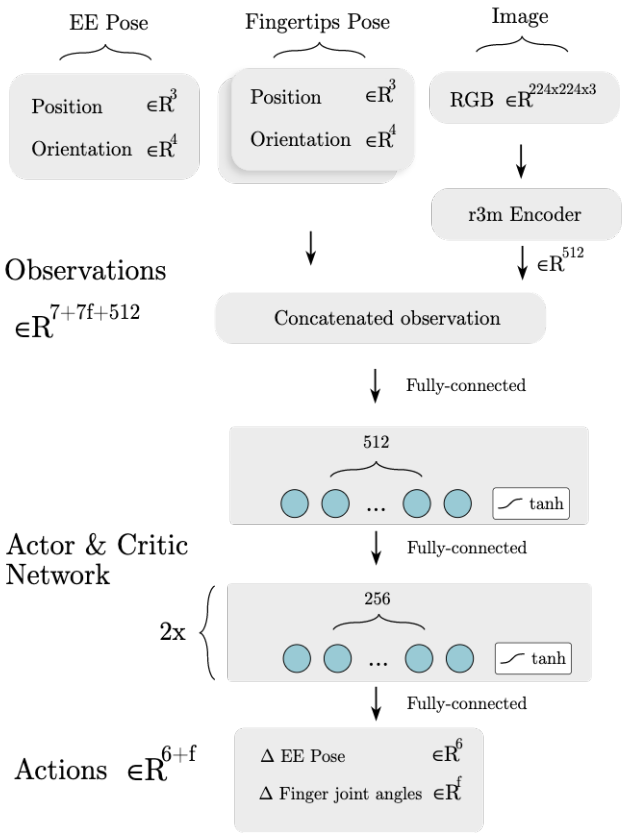


Figure 3.9: Overview of the learning architecture for end-to-end visual observation tasks

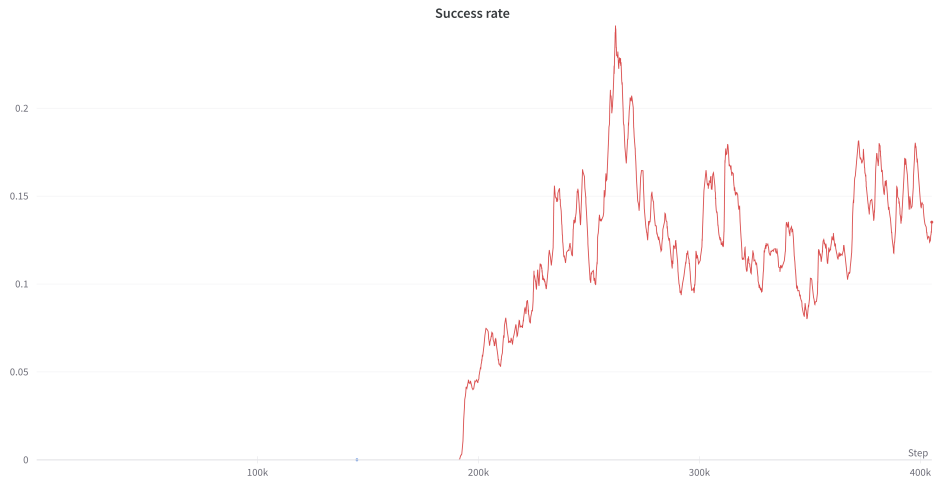


Figure 3.10: Success rate over training steps (equivalent to 14 days of training) for the end-to-end visual-based policy grasping the static lemon object from YCB.

Chapter 4

Conclusion and Outlook

4.1 Conclusion

We developed a training pipeline and a modular training environment including a digital twin of the robotic waste sorting environment. Our work began with single-object static grasping where we saw impressive results, with accuracy rates hitting 97% and 98% for a five-finger gripper and parallel-jaw gripper respectively. Building upon this foundation, we ventured into more complex tasks, training our RL policy to grasp objects on a moving conveyor belt using object pose observations. The policy performed well, achieving a success rate of 87%.
These initial results also led us to explore the realm of visual-based RL for robotic picking and sorting. We implemented and evaluated a visual (RGB) end-to-end static policy for robotic grasping, which not only was able to learn to grasp a single static object using encoded RGB images of the environment with a 97% success rate, but also demonstrated the capability to generalize to unseen objects during training.
Our work demonstrates that deep reinforcement learning (DRL) has great potential for the challenging task of robotic waste sorting. We have shown that robust grasps through pre- and re-grasp manipulations can be learned from low-dimensional representations such as object poses, and compared these representations to bounding boxes and visual embeddings. Moreover, we have proposed a single architecture, observation/action formulation, and reward scheme that can be used to grasp static and dynamic objects of varying complexity, using different morphologies, paving the way for future hardware/software co-design.

4.2 Outlook

As we look toward the future, several avenues of research and development can further improve and extend the contributions of this thesis. These include:

4.2.1 Training framework

Curriculum learning: this is a well-known effective approach to facilitate training in complex tasks by gradually increasing the difficulty of the task, thus allowing the model to build upon its acquired knowledge. In future work, incorporating curriculum learning might be a promising avenue to explore, especially if the task complexity increases or the initialization randomness becomes more pronounced. A well-designed curriculum could enable the learning process to progress more

smoothly by gradually introducing new challenges, allowing the model to adapt and overcome them step by step. This strategy could lead to a more efficient and effective learning process and potentially yield better results in more challenging tasks.
Image Encoder in-the-loop training or fine tuning based on a representative dataset of images from the simulator or the real world.
Sweeps for hardware-software co-design support could be added to also sweep through robot morphologies, sensor placements, etc, so that given a task and a dataset one can easily co-design the optimal hardware and software configuration.

4.2.2 Milestones

Integrate state-space policy into ARC pipeline: By integrating the state-space policy into the Autonomous Recycling Center (ARC) pipeline, we can enhance the overall system's efficiency and capability. This integration would involve passing the scanning stage poses to the RL agent, allowing for seamless interaction between various components of the pipeline.

Visual policy deployment: Deploying the simple visual policy to a real system is a crucial step in bridging the gap between simulation and reality. In future work, we will tackle the sim2real transfer problem, adapting our policies to account for discrepancies between simulated and real-world environments.

Student-Teacher Visual Policy: We propose using state-space policies as teachers for more efficient training of visual policies. By leveraging the knowledge acquired by state-space policies, we can guide the training process of visual policies and accelerate convergence towards robust grasping behavior.

The current framework not only enables the creation of new tasks but also serves as a versatile platform for hardware/software co-design. It allows for rapid design iterations, providing valuable insights before committing to significant investments in physical components. Using the provided example pipelines, one could experiment with different gripper arrangements, sensor modalities, or even swap the robots entirely to explore various combinations and their impact on grasping performance.

Bibliography

[1] M. Koskinopoulou, F. Raptopoulos, G. Papadopoulos, N. Mavrakis, and M. Maniadakis, "Robotic waste sorting technology: Toward a vision-based categorization system for the industrial robotic separation of recyclable waste," *IEEE Robotics Automation Magazine*, vol. 28, no. 2, pp. 50–60, 2021.

[2] T. J. Lukka, T. Tossavainen, J. V. Kujala, and T. Raiko, "Zenrobotics recycler - robotic sorting using machine learning," 2014.

[3] B. S. et al., "Towards robotic waste sorting using deep reinforcement learning," 2022.

[4] D. Strübin and X. Voellmy, "Classification of riverine waste using simulated data," 2022.

[5] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," *CoRR*, vol. abs/1602.00763, 2016.

[6] S. Kumra, S. Joshi, and F. Sahin, "Antipodal robotic grasping using generative residual convolutional neural network," *CoRR*, vol. abs/1909.04810, 2019.

[7] M. Hutter, "Maximising multi-robot waste removal using rl," 2020.

[8] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[10] F. Esquivel, "Robotic sorting of riverine trash," 2021.

[11] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *CoRR*, vol. abs/2103.14030, 2021.

[12] I. Akinola, J. Xu, S. Song, and P. K. Allen, "Dynamic grasping with reachability and motion awareness," *CoRR*, vol. abs/2103.10562, 2021.

[13] S. S. Sajjan, M. Moore, M. Pan, G. Nagaraja, J. Lee, A. Zeng, and S. Song, "Cleargrasp: 3d shape estimation of transparent objects for manipulation," 2019.

[14] T. Kollar, M. Laskey, K. Stone, B. Thananjeyan, and M. Tjersland, "Simnet: Enabling robust unknown object manipulation from pure synthetic data via stereo," 2021.

[15] W. Gao and R. Tedrake, "kpam 2.0: Feedback control for category-level robotic manipulation," 2021.

[16] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3m: A universal visual representation for robot manipulation," 2022.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[18] M. Mosbach and S. Behnke, "Efficient representations of object geometry for reinforcement learning of interactive grasping policies," 2022.

[19] W. Zhou and D. Held, "Learning to grasp the ungraspable with emergent extrinsic dexterity," 2022.

[20] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," 2018.

[21] S. James, K. Wada, T. Laidlow, and A. J. Davison, "Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation," 2022.

[22] S. James and P. Abbeel, "Coarse-to-fine q-attention with learned path ranking," 2022.

[23] ——, "Coarse-to-fine q-attention with tree expansion," 2022.

[24] V. Kumar, D. Hoeller, B. Sundaralingam, J. Tremblay, and S. Birchfield, "Joint space control via deep reinforcement learning," 2021.

[25] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," *CoRR*, vol. abs/2108.10470, 2021.

[26] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," *CoRR*, vol. abs/2109.11978, 2021.

[27] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "Anymal - a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 38–44.

[28] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, P. P. Tehrani, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," 2023.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[30] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12. MIT Press, 1999.

[31] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015.

[32] D. Makoviichuk and V. Makoviychuk, "rl-games: A high-performance framework for reinforcement learning," https://github.com/Denys88/rl_games, May 2021.

[33] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 International Conference on Advanced Robotics (ICAR)*, 2015, pp. 510–517.

[34] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," 2017.

Appendix A

Hardware reference

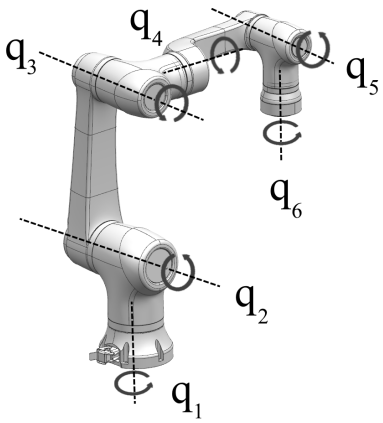


Figure A.1: Joint angle naming and axes orientation of the Lara5 robot.

NEURA ROBOTICS

LARA

Lightweight Agile Robotic Assistant

Data sheet

Combining lightweight design and industrial performance with an intuitive user interface.

LARA, the Lightweight Agile Robotic Assistant, is a six-degree-of-freedom collaborative robot that combines t agility of lightweight design with industrial performance. With unmatched speed, precision and protection, LARA bridges the gap between the world of collaborative and industrial robots. This way, LARA allows you to autom any production process in a cost-efficient way. Its int user interface enables anyone to create programs for LARA, easily automating simple and repetitive tasks i production environment.

General specs	LARA 3	LARA 5	LARA 8	LARA 10	LARA 15
Payload	3 kg	5 kg	8 kg	10 kg	15 kg
Reach	590 mm	800 mm	1300 mm	1000 mm	1300 mm
Degrees of Freedom	6 rotary joints	6 rotary joints	6 rotary joints	6 rotary joints	6 rotary joints
Weight	17 kg	26 kg	48 kg	42 kg	55 kg
Robot mounting	Any orientation	Any orientation	Any orientation	Any orientation	Any orientation
IP classification	IP66	IP66	IP66	IP66	IP54
Ambient working temperature	0 °C – 50 °C	0 °C – 50 °C	0 °C – 50 °C	0 °C – 50 °C	0 °C – 50 °C
Data & power cables	Complete inner harness	Complete inner harness	Complete inner harness	Complete inner harness	Complete inner harness
Footprint base	Ø 156 mm	Ø 156 mm	Ø 200 mm	Ø 200 mm	Ø 200 mm
Tool connector type	M12 12-pole	M12 12-pole	M12 12-pole	M12 12-pole	M12 12-pole
Status illumination	RGB LED on TCP	RGB LED on TCP	RGB LED on TCP	RGB LED on TCP	RGB LED on TCP
Tgt. perf. level	PLd Cat.3 / SIL3	PLd Cat.3 / SIL3	PLd Cat.3 / SIL3	PLd Cat.3 / SIL3	PLd Cat.3 / SIL3
Tgt. repeatability	± 0.02 mm	± 0.02 mm	± 0.02 mm	± 0.02 mm	± 0.02 mm

Designed and engineered in Germany

Robotiq 2f-140:

Specifications

	Hand-E		2F-85		2F-140	
Stroke	50 mm	2.0 in	85 mm	3.3 in	140 mm	5.5 in
Grip Force	20 to 130 N	4.5 to 27 lbf	20 to 235 N	4.5 to 50 lbf	10 to 125 N	2 to 25 lbf
Form-fit Grip Payload	5 kg	11 lbs	5 kg	11 lbs	2.5 kg	5.5 lbs
Friction Grip Payload*	3 kg	7 lbs	5 kg	11 lbs	2.5 kg	5.5 lbs
Gripper Weight	1 kg	2 lbs	0.9 kg	2 lbs	1 kg	2 lbs
Closing speed	20 to 150 mm/s	0.8 to 5.9 in/s	20 to 150 mm/s	0.8 to 5.9 in/s	30 to 250 mm/s	1.2 to 9.8 in/s
Ingress protection (IP) rating	IP67		IP40		IP40	

*Calculated for the use of silicon covered fingertips to grip a steel object, at a low robot acceleration.

Intel RealSense D455:

Intel® RealSense™ Depth Family Offerings

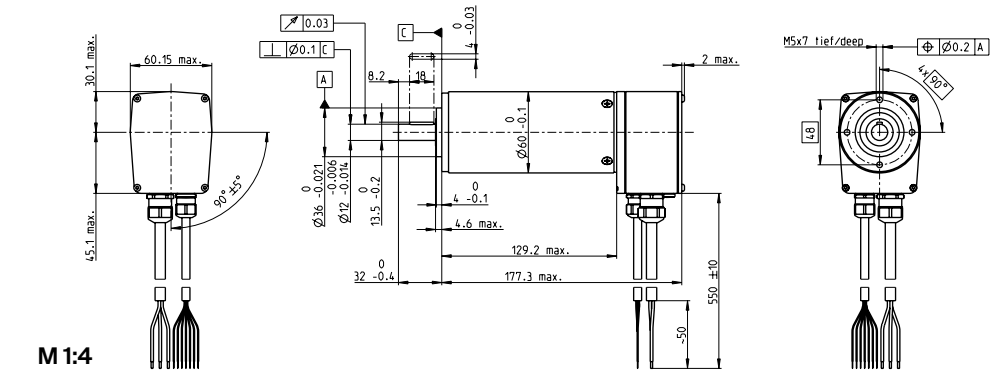
	D415	D435/D435i	D455	SR305	L515
Use Case	3D scanning, facial authentication, robotics: collision avoidance and pick and place, recognition and interaction use cases	3D scanning, facial authentication, robotics: collision avoidance and pick and place, recognition and interaction use cases	3D scanning, robotics: collision avoidance, recognition and interaction use cases	Indoor short range objects with small details, Facial Recognition, gesture (hand tracking), close in scanning	Cases where scanning accuracy is paramount and lighting is controlled. Box measurement, object, and room scanning
Value	Multicamera in same FOV; Best priced stereo camera	Global shutter with wide FOV; Multicamera in same FOV	Global shutter with wide FOV; Multicamera in same FOV; Match FOV Global shutter RGB with depth and HW sync trigger	Coded light solution with good accuracy at short range; Excellent price	High Accuracy over entire range; world's smallest hi-res LiDAR camera
Optimal Range (Stereo cameras can see further out – Distance – accuracy worsens geometrically)	.4m to 3m	.4m to 3m	.4m to 6m	.2m to 1.5m (accuracy is ~consistent)	.4m to 9m (accuracy is ~consistent)
Z Error < 2% Range and Min Z @ Max Resolution	<2% at 2m Min Z ~50 cm	<2% at 2m Min Z ~30cm	<2% at 4m Min Z ~40cm	<3% at 1.5m Min Z ~20cm	~3mm to ~16mm thru 9m Min Z ~25cm
Environment	Indoor & Outdoor	Indoor & Outdoor	Indoor & Outdoor	Indoors	Indoors
Max Depth Resolution	1280x720 @ 30fps 848x480 @ 90fps	1280x720 @ 30fps 848x480 @ 90fps	1280x720 @ 30fps 848x480 @ 90fps	640x480 @ 60fps	1024x768 @ 30fps
Depth FOV HD/Shutter[1]	64°x41°/Rolling	86°x57°/Global	86°x57°/Global	65°x50°/Global	70°x55°
RGB/Shutter[1]	2MP/64°x41°/Rolling	2MP/64°x41°/Rolling	1MP (matched to depth FOV)/Global	2MP/Rolling	2MP/Rolling
Product Dimensions (mm)	99x23x20	90x25x25	124x29x26	139x26x12	61 diameter x 26
IMU	NO	NO (D435i YES)	YES	NO	YES

[1] FOV is measured +/-3° of stated value
* Final technical specs not available until QS for each product

intel REALSENSE

EC

EC 60 ∅60 mm, bürstenlos, 400 Watt



M 1:4

Lagerprogramm
Standardprogramm
Sonderprogramm (auf Anfrage)

Artikelnummern

Motordaten			167132	167131
	Werte bei Nennspannung			
1	Nennspannung	V	48	48
2	Leerlaufdrehzahl	min⁻¹	5370	3100
3	Leerlaufstrom	mA	670	268
4	Nenndrehzahl	min⁻¹	4960	2680
5	Nennmoment (max. Dauerdrehmoment)	mNm	768	843
6	Nennstrom (max. Dauerbelastungsstrom)	A	9.56	5.9
7	Anhaltemoment	mNm	11800	6820
8	Anlaufstrom	A	139	46.4
9	Max. Wirkungsgrad	%	87	86
	Kenndaten			
10	Anschlusswiderstand Phase-Phase	Ω	0.345	1.03
11	Anschlussinduktivität Phase-Phase	mH	0.273	0.82
12	Drehmomentkonstante	mNm/A	84.9	147
13	Drehzahlkonstante	min⁻¹/V	113	65
14	Kennliniensteigung	min⁻¹/mNm	0.457	0.457
15	Mechanische Anlaufzeitkonstante	ms	3.98	3.98
16	Rotorträgheitsmoment	gcm²	831	831

Spezifikationen

Thermische Daten

17	Therm. Widerstand Gehäuse-Luft	1.3 K/W
18	Therm. Widerstand Wicklung-Gehäuse	0.5 K/W
19	Therm. Zeitkonstante der Wicklung	33.9 s
20	Therm. Zeitkonstante des Motors	1200 s
21	Umgebungstemperatur	-20...+100°C
22	Max. Wicklungstemperatur	+125°C

Mechanische Daten (vorgespannte Kugellager)

23	Grenzdrehzahl	7000 min⁻¹
24	Axialspiel bei Axiallast < 30 N	0 mm
	> 30 N	max. 0.14 mm
25	Radialspiel	vorgespannt
26	Max. axiale Belastung (dynamisch)	24 N
27	Max. axiale Aufpresskraft (statisch)	392 N
	(statisch, Welle abgestützt)	6000 N
28	Max. radiale Belastung, 5 mm ab Flansch	240 N

Weitere Spezifikationen

29	Polpaarzahl	1
30	Anzahl Phasen	3
31	Motorgewicht	2450 g
	Schutzgrad	IP54*

Motordaten gemäss Tabelle sind Nenndaten.

Anschlüsse Motor (Kabel AWG 16)

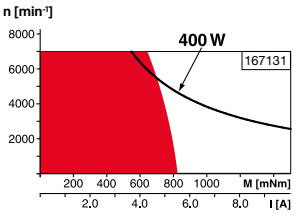
Kabel 1	Motorwicklung 1
Kabel 2	Motorwicklung 2
Kabel 3	Motorwicklung 3

Anschlüsse Sensoren (Kabel AWG 24)[1)]

weiss	Hall-Sensor 3
braun	Hall-Sensor 2
grün	Hall-Sensor 1
gelb	GND
grau	V_{Hall} 4.5 ... 24 VDC
blau	Temperatursensor (PTC)
rosa	Temperatursensor (PTC)

[1)] In Kombination mit Resolver nicht herausgeführt.
Temperaturüberwachung, PTC Widerstand
Micropille 110°C, R 25°C < 0.7 kΩ,
R 115°C ≥ 2.66 kΩ, R 125°C ≥ 8.0 kΩ
Schaltbild für Hall-Sensoren siehe S. 47

Betriebsbereiche



Legende

Dauerbetriebsbereich
Unter Berücksichtigung der angegebenen thermischen Widerstände (Ziffer 17 und 18) und einer Umgebungstemperatur von 25°C wird bei dauernder Belastung die maximal zulässige Rotortemperatur erreicht = thermische Grenze.

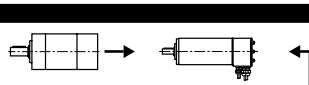
Kurzzeitbetrieb
Der Motor darf kurzzeitig und wiederkehrend überlastet werden.

Typenleistung

maxon Baukastensystem

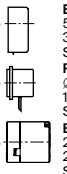
Details auf Katalogseite 36

Planetengetriebe
∅81 mm
20 - 120 Nm
Seite 404



Empfohlene Elektronik:

Hinweise	Seite 36
ESCON Mod. 50/4 EC-S	487
ESCON Mod. 50/8 (HE)	488
ESCON 70/10	489
DEC Module 50/5	491
EPOS4 Module 50/15	497
EPOS4 Module 50/8	497
EPOS4 Comp. 50/8 CAN	499
EPOS4 Comp. 50/15 CAN	500
EPOS4 70/15	501



Encoder HEDL 9140
500 Imp.,
3 Kanal
Seite 478

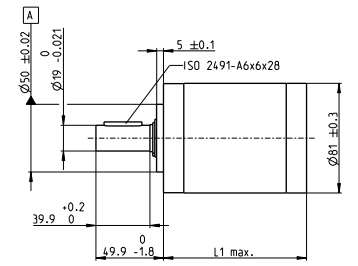
Resolver Res
∅26 mm
10 V
Seite 481

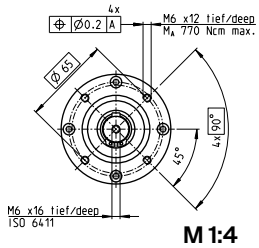
Bremse AB 41
24 VDC
2.0 Nm
Seite 523

*Schutzgrad nur in eingebautem Zustand mit Abdichtung flanschseitig.

gear

Planetengetriebe GP 81 A ∅81 mm, 20.0–120.0 Nm



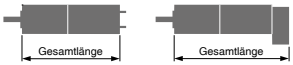


M 1:4

Technische Daten			
Planetengetriebe			geradeverzahnt
Abtriebswelle			Stahl
Abtriebswellenlagerung			Kugellager
Radialspiel, 8 mm ab Flansch			max. 0.1 mm
Axialspiel			max. 1 mm
Max. axiale Aufpresskraft			1500 N
Drehsinn, Antrieb zu Abtrieb			=
Max. Eingangsdrehzahl dauernd			3000 min^{-1}
Empfohlener Temperaturbereich			-30...+140°C
Stufenzahl	1	2	3
Max. radiale Belastung, 24 mm ab Flansch	400 N	600 N	1000 N
Max. axiale Belastung (dynamisch)	80 N	120 N	200 N

Lagerprogramm
Standardprogramm
Sonderprogramm (auf Anfrage)

Artikelnummern						
	110408	110409	110410	110411	110412	110413
Getriebedaten						
1 Untersetzung	3.7:1	14:1	25:1	51:1	93:1	308:1
2 Untersetzung absolut	$^{63}/_{17}$	$^{3969}/_{289}$	$^{1701}/_{68}$	$^{250047}/_{4913}$	$^{107163}/_{1156}$	$^{19683}/_{64}$
3 Max. Motorwellendurchmesser mm	14	14	14	14	14	14
4 Stufenzahl	1	2	2	3	3	3
5 Max. Dauerdrehmoment Nm	20	60	60	120	120	120
6 Kurzzeitig zulässiges Drehmoment Nm	30	90	90	180	180	180
7 Max. Wirkungsgrad %	80	75	75	70	70	70
8 Gewicht g	2300	3000	3000	3700	3700	3700
9 Mittleres Getriebespiel unbelastet °	0.5	0.55	0.55	0.6	0.6	0.6
10 Massenträgheitsmoment gcm^2	165	155	125	88	154	89
11 Getriebelänge L1 mm	92.0	113.7	113.7	135.3	135.3	135.3

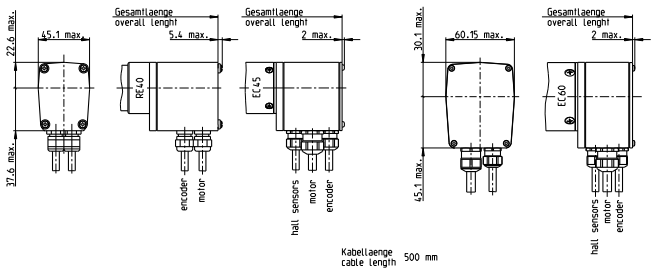


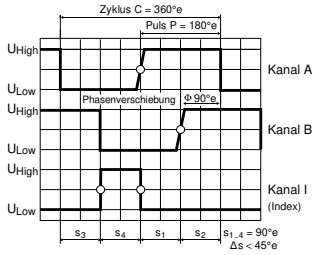
maxon Baukastensystem

+ Motor	Seite	+ Sensor	Seite	Bremse	Seite	Gesamtlänge [mm] = Motorlänge + Getriebelänge + (Sensor/Bremse) + Montageteile					
RE 65, 250 W	143					223.5	245.2	245.2	266.8	266.8	266.8
RE 65, 250 W	143	HEDS 5540	472			249.4	271.1	271.1	292.7	292.7	292.7
RE 65, 250 W	143	HEDL 5540	474			249.4	271.1	271.1	292.7	292.7	292.7
RE 65, 250 W	143	HEDL 9140	479			279.6	301.3	301.3	322.9	322.9	322.9
RE 65, 250 W	143			AB 44	524	279.6	301.3	301.3	322.9	322.9	322.9
RE 65, 250 W	143	HEDL 9140	479	AB 44	524	297.6	319.3	319.3	340.9	340.9	340.9
EC 60, 400 W	232					269.4	291.1	291.1	312.7	312.7	312.7
EC 60, 400 W	232	HEDL 9140	478			269.4	291.1	291.1	312.7	312.7	312.7
EC 60, 400 W	232	Res 26	481			269.4	291.1	291.1	312.7	312.7	312.7
EC 60, 400 W	232			AB 41	523	283.0	304.7	304.7	326.3	326.3	326.3
EC 60, 400 W	232	HEDL 9140	478	AB 41	523	307.0	328.7	328.7	350.3	350.3	350.3

sensor

Encoder HEDL 9140 500 Impulse, 3 Kanal, mit Line Driver RS 422





Drehrichtung cw (Definition cw S. 68)

Lagerprogramm
Standardprogramm
Sonderprogramm (auf Anfrage)

Typ	Artikelnummern
	137959
Impulszahl pro Umdrehung	500
Anzahl Kanäle	3
Max. Impulsfrequenz (kHz)	100
Max. Drehzahl (min⁻¹)	12 000

maxon Baukastensystem

+ Motor	Seite	+ Getriebe	Seite	+ Bremse	Seite	Gesamtlänge [mm], siehe Getriebe
RE 40, 150 W	141					125.1
RE 40, 150 W	141	GP 42, 3 - 15 Nm	396			○●
RE 40, 150 W	141	GP 52, 4 - 30 Nm	401			○●
RE 40, 150 W	141			AB 28	520	135.6
RE 40, 150 W	141	GP 42, 3 - 15 Nm	396	AB 28	520	○●
RE 40, 150 W	141	GP 52, 4 - 30 Nm	401	AB 28	520	○●
EC 45, 150 W	230					126.8
EC 45, 150 W	230	GP 42, 3 - 15 Nm	396			○●
EC 45, 150 W	230	GP 52, 4 - 30 Nm	401			○●
EC 45, 150 W	230			AB 28	520	135.6
EC 45, 150 W	230	GP 42, 3 - 15 Nm	396	AB 28	520	○●
EC 45, 150 W	230	GP 52, 4 - 30 Nm	401	AB 28	520	○●
EC 45, 250 W	231					159.6
EC 45, 250 W	231	GP 42, 3 - 15 Nm	397			○●
EC 45, 250 W	231	GP 52, 4 - 30 Nm	401			○●
EC 45, 250 W	231	GP 62, 8 - 50 Nm	403			○●
EC 45, 250 W	231			AB 28	520	168.4
EC 45, 250 W	231	GP 42, 3 - 15 Nm	396	AB 28	520	○●
EC 45, 250 W	231	GP 52, 4 - 30 Nm	401	AB 28	520	○●
EC 45, 250 W	231	GP 62, 8 - 50 Nm	403	AB 28	520	○●
EC 60, 400 W	232					177.3
EC 60, 400 W	232	GP 81, 20 - 120 Nm	404			○●
EC 60, 400 W	232			AB 41	523	214.9
EC 60, 400 W	232	GP 81, 20 - 120 Nm	404	AB 41	523	○●

Technische Daten

Versorgungsspannung V_{cc}	5 V ± 10%
Typische Stromaufnahme	55 mA
Ausgangssignal	EIA Standard RS 422
verwendeter Treiber:	DS26LS31
Phasenverschiebung Φ	90°e ± 45°e
Signalanstiegszeit (typisch, bei C_L = 25 pF, R_L = 1 kΩ, 25 °C)	180 ns
Signalabfallzeit (typisch, bei C_L = 25 pF, R_L = 1 kΩ, 25 °C)	40 ns
Indexpulsbreite	90°e
Betriebstemperaturbereich	-40...+85 °C
Trägheitsmoment der Impulsscheibe	≤0.6 gcm²
Max. Winkelbeschleunigung	250 000 rad s⁻²
Strom pro Kanal	± 20 mA

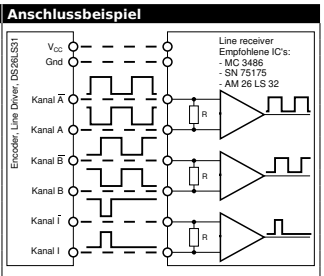
Das Indexsignal ist synchronisiert mit Kanal A und B.

Kabelbelegung

Kabel weiss	=	2	V_{CC} 5 VDC
Kabel braun	=	3	GND
Kabel grün	=	5	Kanal $\overline{A}$
Kabel gelb	=	6	Kanal A
Kabel grau	=	7	Kanal $\overline{B}$
Kabel rosa	=	8	Kanal B
Kabel blau	=	9	Kanal $\overline{I}$ (Index)
Kabel rot	=	10	Kanal I (Index)

Kabelquerschnitt 8 × 0.25 mm²

Anschlussbeispiel



Abschlusswiderstand R = typisch 120 Ω