Bringing the RT-1-X Foundation Model for Robotic Control to New Embodiments

Jonathan Salzer
jonathan@salzer.net

August 28, 2024, 74 pages

Academic supervisor: Dr. Arnoud Visser, a.visser@uva.nl
Research group: Intelligent Robotics Lab, www.intelligentroboticslab.nl

Universiteit van Amsterdam
Faculteit der Natuurwetenschappen, Wiskunde en Informatica
Master Software Engineering
http://www.software-engineering-amsterdam.nl

Abstract

Traditional robotic systems require specific training data for each task, environment, and robot form (known as embodiment). While recent advancements in machine learning have enabled models to generalize across new tasks and environments, the challenge of adapting these models to entirely new embodiments remains largely unexplored. This master's thesis addresses this by investigating the generalization capabilities of the RT-1-X robotic foundation model to embodiments unseen during its training.

The research begins with a comprehensive series of zero-shot experiments on a new embodiment, followed by fine-tuning the model on a small dataset specific to the target embodiment, and comparing the resulting performance. The findings reveal that RT-1-X does not generalize zero-shot to the tested embodiment. However, fine-tuning significantly improves performance, enabling faster convergence during training and better outcomes compared to training from scratch on the same dataset. Despite these improvements, the model still fails to transfer skills or object knowledge from the pre-training dataset to the new embodiment.

These results demonstrate that even state-of-the-art foundation models trained on a diverse set of embodiments are unable to generalize to new embodiments in a zero-shot setting. While fine-tuning enhances adaptability, substantial limitations persist.

Contents

Chapter 1

Introduction

Recent breakthroughs in machine learning and artificial intelligence suggest that training on large, diverse datasets can lead to highly adaptable models, which often exceed the performance of models developed for specific tasks using smaller datasets [1]. Traditional approaches to robotic leaning have required task-specific datasets tailored to each individual task, environent, and robot, limiting the scalability and flexibility of robotic applications. As a result, the field has been exploring more generalizable models that can adapt to new scenarios with minimal retraining. Recent advancements like Transformer architectures and robotic foundation models have led to models like Google's RT-1 [2], which demonstrate the potential for robots to generalize across various tasks and environments. However, one critical area remains underexplored: the ability of these models to generalize across entirely new robotic embodiments.

This thesis addresses this gap by investigating the cross-embodiment generalization capabilities of the RT-1-X model. By implementing this model on a new robotic platform—the UMI-RTX robot at the University of Amsterdam's Intelligent Robotics Lab—this research seeks to understand how well the model can transfer its learned knowledge to a robot with different physical characteristics and operational contexts. Both zero-shot generalization, as well as adaptation to the new embodiment with minimal fine-tuning, are explored.

The findings show that zero-shot generalization to the new embodiment is not possible. By fine-tuning the model on a small dataset from the target embodiment, generalization could be significantly improved, although still not reaching the performance achieved on the original embodiments.

1.1 Problem statement

One of the most significant challenges in the field of robotics is the acquisition of suitable datasets for training machine learning models [1]. Traditional approaches to robotics require large amounts of data tailored to each specific robot, task, and environment. This data is often expensive and time-consuming to create, as it typically involves engineering-heavy autonomous operations or costly human demonstrations. Furthermore, because of the vast diversity in robotic embodiments—each with unique physical characteristics and operational contexts—datasets are rarely reusable across different robots. This limitation hampers the scalability and adaptability of robotic systems, making it difficult to develop versatile models that can operate effectively across various settings without extensive retraining. [1]

The high cost and limited reusability of robotic datasets have driven the search for more generalizable models that can be adapted to new environments, tasks, and, crucially, new robotic embodiments with minimal additional effort. Significant advancements in machine learning have recently enabled a paradigm shift across various domains, including computer vision and natural language processing (NLP), by moving away from highly specific datasets and models toward more general models based on broad, task-agnostic datasets. These high-capacity foundation models, pre-trained on large-scale, diverse datasets, provide a robust platform for a wide range of downstream tasks, such as semantic reasoning, problem solving, and visual interpretation. The ability of these models to generalize across tasks and environments without requiring task-specific datasets offers tremendous advantages in robotics, where the generation of datasets is even more costly than in other domains.

Recently, Google's RT-1 model has exemplified a significant shift toward generalization in robotics. Leveraging large-scale, task-agnostic datasets and an efficient Transformer architecture, RT-1 has demonstrated remarkable capabilities in generalizing to new tasks, objects, and environments that were not part of its original training dataset. Another major advancement in this area has been the introduction

of the Open X-Embodiment dataset [1], the first of its kind due to its unprecedented diversity in robotic embodiments. This dataset combines a vast number of embodiment-specific datasets, encompassing over one million robot trajectories from 22 different robots. The diversity and scale of the Open X-Embodiment dataset have significantly enhanced the RT-1 model, enabling it to outperform traditional task-specific models by wide margins and showcasing its potential for broad application across various tasks and environments [1].

However, while the RT-1 model in combination with the Open X-Embodiment dataset has shown considerable success in generalizing across unseen tasks, objects, and environments, the challenge of generalizing to entirely new robotic embodiments has been much less explored. Generalizing across embodiments is particularly complex because each robot can have different physical structures, sensors, and modes of interaction with the environment. If a model can effectively transfer its learned knowledge to a completely new robot embodiment without significant degradation in performance, it would reduce the need for extensive, embodiment-specific datasets and enable more flexible deployment of robots across different platforms, thereby lowering costs and expanding the applicability of robotic systems. On a larger scale, advancing embodied AI in this way is a crucial step toward achieving artificial general intelligence, as it comes closer to developing systems that can adapt and function effectively in a wide range of physical contexts, much like humans do [2].

Currently, there is a significant gap in our understanding of how these foundation models perform when faced with new robotic embodiments. While the X-Embodiment dataset includes a variety of robots, the ability of these models to generalize to embodiments not included in the training set has not been empirically demonstrated. This leaves open critical questions about the practicality of using these models in real-world scenarios where robots might differ significantly from those in the training data.

This master's thesis aims to address this gap by focusing on the generalization capabilities to unseen embodiments. Specifically, the research will involve implementing and evaluating the RT-1-X foundation model on a new, previously unseen robotic embodiment—the UMI-RTX robot at the University of Amsterdam's Intelligent Robotics Lab. The primary objective is to investigate the model's ability to adapt to this new embodiment, which was not part of its original training dataset, and to identify any potential performance trade-offs or limitations that may arise.

By exploring the model's ability to generalize to a completely new embodiment, this thesis will contribute to the understanding of robotic foundation models and their adaptability to diverse hardware platforms. The findings from this research contribute valuable insights into developing more adaptable models that could potentially be deployed across various robotic systems with reduced need for extensive retraining. This work aims to support the ongoing advancement of foundation models in robotics, helping to facilitate their broader application and enabling the use of advanced robotic technologies in a wider range of domains.

1.1.1 Research questions

Building on the challenges identified in the problem statement, this thesis aims to explore the generalization capabilities of the RT-1-X model, particularly its ability to adapt to new robotic embodiments. To structure this investigation, this thesis focuses on two central research questions:

RQ1: Can the RT-1-X model generalize to a completely unseen robotic embodiment, without any additional data, utilizing knowledge learned from the Open X-Embodiment dataset?

This question investigates whether the RT-1-X model can effectively transfer its learned knowledge from the diverse set of robotic embodiments in the Open X-Embodiment dataset to a new, previously unseen robot without requiring any additional training or data. The goal is to assess the model's ability to perform tasks on the UMI-RTX robot solely based on its prior training, thereby evaluating the feasibility of zero-shot generalization to new embodiments.

Building on the findings of RQ1, the second research question explores the potential for improving the model's performance through fine-tuning:

RQ2: Can fine-tuning RT-1-X on a small number of demonstrations from a new robotic embodiment improve its performance, and what are the benefits of using a pre-trained model?

To get a nuanced understanding of the different performance aspects, this question is further divided into three subquestions:

- **RQ2.1: How does the fine-tuned RT-1-X model perform on the specific task for which it was fine-tuned?**
  This subquestion focuses on the in-distribution performance of the fine-tuned model, evaluating its effectiveness on the task that was included in the fine-tuning dataset. The objective is to understand how well the model can learn and execute a specific task on the new embodiment.
- **RQ2.2: Can the fine-tuned RT-1-X model transfer concrete knowledge learned during Open X-Embodiment pre-training to the new embodiment and maintain performance when faced with changes in the demonstration environment?**
  This subquestion examines the model's ability to transfer concrete skills learned during the Open X-Embodiment pre-training step to the fine-tuning domain. It also assesses whether the fine-tuned model can maintain performance when exposed to environment factors that were unseen in fine-tuning.
- **RQ2.3: What benefits does pre-training on the Open X-Embodiment dataset provide when fine-tuning RT-1-X on a new embodiment?**
  This subquestion investigates the advantages of using a pre-trained foundation model as a basis for fine-tuning, instead of training from scratch for the new embodiment. It seeks to determine how Open X-Embodiment pre-training impacts the model's ability to adapt to the new embodiment and improve its overall performance.

1.1.2 Research Method

The research conducted in this thesis follows a systematic approach designed to explore the cross-embodiment generalization capabilities of the RT-1-X model step-by-step. The process began with setting up the necessary components and gradually moved toward more complex experimentation and analysis. The methodology was designed to address the research questions, with each step building on the previous one.

Component Preparation and System Integration

The research began with the necessary preparatory work to ensure that the essential components—the UMI robot and the RT-1-X model—were operational. Integrating these two components proved to be more challenging than anticipated. The UMI robot, in particular, presented unexpected issues that required significant effort to resolve. This included extensive bug fixing and quality work in the control code to ensure reliable operation. Hardware challenges that occurred due to the age of the robot also needed to be addressed to achieve full functionality throughout the project. On the RT-1-X side, the code from the original open-source repository [3] — designed as a basic inference demonstration with dummy data — had to be developed into a complete inference loop using real inputs from the UMI environment. Finally, integrating RT-1-X with the UMI robot presented its own set of challenges, as it was necessary to ensure that the model's outputs could be accurately interpreted and executed by the robot, taking into account the unique characteristics and intricacies of the UMI platform.

As the final step of system integration, the full pipeline was validated to ensure that the model's behavior was not influenced by any factors introduced during this stage, which could distort the experimental results and compromise their accuracy, representativeness and reproducibility.

Once these components were fully functional and integrated, the stage was set for the first set of experiments, addressing the first main research question.

Experiments addressing RQ1

With the UMI robot and RT-1-X model fully operational after overcoming initial integration challenges, the next step was to conduct a comprehensive series of experiments focused on investigating the model's zero-shot performance on the new robotic embodiment, directly addressing **RQ1**. The primary objective was to determine whether RT-1-X could generalize to the UMI robot without any additional training,

leveraging the knowledge it had acquired from the Open X-Embodiment dataset.

These experiments systematically altered various factors, including task description, target object, workspace layout, but also technical aspects like model output interpretation, to assess their impact on the model's performance and behavior. The aim was to comprehensively explore how these variables influenced the model's behavior and to assess the feasibility of achieving zero-shot transfer with the RT-1-X model on the new embodiment. This approach was designed to thoroughly investigate the model's capabilities across a range of environmental and technical conditions, seeking to understand any potential for generalization to the UMI robot.

UMI Dataset Creation and Model fine-tuning

Building on the insights gained from the zero-shot experiments, the next stage of the research was focused on preparing everything needed to answer RQ2. This involved adapting the RT-1-X model to the UMI robot through fine-tuning and preparing the necessary baseline models for comparison.

To begin, a method for teleoperating the UMI robot was developed, which enabled the collection of a new demonstration dataset. This dataset consists of 100 demonstrations of one simple task performed by the UMI robot. A fine-tuning pipeline for RT-1-X was then built and validated, similar to the system integration stage, to ensure that the model's behavior was not distorted due to training issues. RT-1-X was fine-tuned on the UMI dataset to adapt it to the specific characteristics of the UMI robot.

In addition to fine-tuning RT-1-X, other models necessary for answering RQ2 were also trained. This included training RT-1 from scratch on the UMI dataset, and fine-tuning another Open X-Embodiment based foundation model on the UMI dataset. The training pipelines for these models was developed, and again validated. These preparations ensured that all required models were ready for the subsequent experiments aimed at evaluating the performance and capabilities of RT-1-X in the context of the new robotic embodiment.

Experiments Addressing RQ2

With the RT-1-X model now fine-tuned on the UMI dataset and ready for inference, the next phase involved conducting a series of experiments to systematically address the subquestions of Research Question 2 (RQ2). These experiments were designed to evaluate the model's performance across several key dimensions:

Addressing RQ2.1: In-Distribution Performance. The first set of experiments was conducted to assess the fine-tuned model's performance on the specific task it was trained on during fine-tuning. This involved running multiple trials where the UMI robot, guided by the fine-tuned RT-1-X model, executed the task under controlled conditions. Performance metrics such as success rate and accuracy were recorded and analyzed to determine how effectively the model could execute the task within the new embodiment after the fine-tuning process.

Addressing RQ2.2: Knowledge Transfer and Generalization. Following the evaluation of in-distribution performance, the experiments shifted focus to assessing the model's ability to transfer knowledge from the pre-training phase to the UMI embodiment. This was done by selecting specific tasks and objects that were present during the Open X-Embodiment pre-training, but not in fine-tuning, and testing the model's performance on these elements with the UMI robot. Additionally, environmental alterations were introduced to evaluate how the performance on the in-distribution task is impacted by these changes.

Addressing RQ2.3: Ablation Studies and Comparisons The final set of experiments involved ablation studies and baseline comparisons to isolate and examine the effects of specific components of the RT-1-X model. In these experiments, the performance of the fine-tuned RT-1-X model was compared with that of baseline models, including the RT-1 model trained from scratch on the UMI dataset and another architecture pre-trained on the Open X-Embodiment dataset and fine-tuned on UMI. Each model was subjected to the same tasks and environmental conditions, with performance metrics carefully recorded and analyzed. This comprehensive evaluation aimed to identify the specific benefits of pre-training on the Open X-Embodiment dataset and to understand the influence of the RT-1 architecture on the model's overall performance.

1.2 Contributions

Besides the experimental results, this research makes the following contributions:

1. A ROS2 package for running inference using RT-1-X [4]
2. A fine-tuning pipeline for adapting RT-1-X to new Embodiments [5]
3. Updated control code for the UMI robot [6] and a dataset of demonstrations with the UMI robot, along with training checkpoints of RT-1 and RT-1-X[1]
4. A contribution to the author's original training code, addressing an issue found during research[2]

1.3 Outline

Chapter 2 equips the reader with the necessary background, starting with a general introduction to robotic learning, followed by a presentation of the three main components this thesis is based on: The RT-1 Transformer model, the Open X-Embodiment dataset, and the UMI robot embodiment. Chapter 3 describes the process of setting up these components, both individually, as well as the required infrastructure to bring them together for this thesis. During the preparation of the individual components, several problems were encountered, which are also described here. Chapter 4 deals with the evaluation of the RT-1-X model on the unseen UMI robot: It is described in detail which experiments were conducted, how they were conducted, and why. In Chapter 5, the process of fine-tuning the RT-1-X model for the UMI robot is discussed, explaining how the UMI dataset was collected and the fine-tuning pipeline was built. This chapter also describes the experiments conducted with the fine-tuned model, as well as with related baseline models that were necessary to get performance comparisons. Chapter 6 finally presents the results of the evaluation of both the zero-shot model, as well as the model fine-tuned for the UMI robot. The implications of these results is discussed in Chapter 7. This chapter interprets the findings in the context of the research questions, providing answers based on the experimental data. It also considers the broader implications of the results for the field of robotic learning and model generalization. Additionally, it discusses additional findings made along the way that, while not directly related to the research questions, are still relevant. Chapter 8 explores related works in the field. It places the research conducted in this thesis within the broader context of robotic learning, offering an overview of other relevant approaches and comparing the results obtained here with those reported in the literature. Additionally, this chapter provides perspectives on alternative methods and emerging trends, broadening the scope of the discussion and highlighting potential future directions for research. Finally, the work is concluded in Chapter 9, where the most important findings are summed up and an outlook on future work is given.

[1] https://drive.google.com/drive/folders/1HYkzqaRuEKcKRUdfXVvH1UKBjGdKGZgf?usp=sharing
[2] https://github.com/google-deepmind/open_x_embodiment/pull/84

Chapter 2

Theoretical Background

Before starting with the actual research, this chapter provides an introduction to the key concepts and prior work necessary to understand the context of this master's thesis. It begins with definitions of the terminology that will be used throughout the thesis, followed by an overview of fundamental concepts in (robotic) machine learning. Finally, the chapter introduces the three core components that this research is based on: the RT-1 model, the Open X-Embodiment dataset, and the UMI robot.

2.1 Terminology

This section aims to introduce the machine learning concepts that are commonly used in modern robotics research. It starts by defining some basic terms that will be used throughout this thesis:

- Embodiment: The physical form or structure of a robot, including its sensors and actuators, which defines how it interacts with its environment. Different robots have different embodiments, leading to variations in how they perceive the world and act upon it.
- State: The current situation or condition of the robot and its environment, as perceived through its sensors. The state includes all relevant information that the robot can use to make decisions.
- Action: A specific command or set of commands that the robot can execute to change its state or interact with its environment, such as moving an arm or adjusting a gripper.
- Trajectory: The sequence of states and actions taken by the robot throughout an episode. A trajectory represents the path that the robot follows from the beginning to the end of an episode.
- Policy: A strategy or function that maps states to actions. In robotic learning, a policy determines what action the robot should take when it encounters a specific state.
- Inference: The process of applying a policy to a state to decide the next action. Inference is how the robot makes decisions in real-time based on its learned policy.
- Episode: A complete sequence of interactions between the robot and its environment, starting from an initial state and ending when a specific goal is reached or a stopping condition occurs. An episode consists of multiple steps where the robot state is observed, an action is taken (either by the policy during inference, or by a human during manual control), and the embodiment transitions to a new state.

2.2 Machine Learning in Robotics

This section introduces machine learning in the domain of robotics. The topic of machine learning is extremely broad and highly complex, and there are many different fields of robotics where it has been applied. The domain is also constantly expanding, with new work published very regularly. To stay within the scope, this introduction will be organized around RT-1-X and its position within the robotic learning field.

2.2.1 End-to-end Models

Machine learning has had a significant impact on various areas of robotics, including computer vision, motion planning, and manipulation. Traditionally, robotic systems have relied on modular approaches

where each component—such as vision, planning, and grasping—is handled separately, often using a combination of deterministic algorithms and specialized machine learning models [7–9]. This modular approach, while effective in many scenarios, can present challenges in integration, as each module must be individually trained, carefully tuned to work together, and errors in one module can propagate through the system, potentially degrading overall performance.

In contrast, **end-to-end learning** in robotics has emerged as a powerful alternative. In this approach, raw sensor data from the robot (e.g. camera images) is used as input, and the desired robotic actions are directly issued by the model. This method simplifies the system architecture by eliminating the need for separate, hand-engineered modules, allowing the model to learn a direct mapping from perception to action. [10, 11]

One of the primary challenges is the requirement for large amounts of labeled data to train the model effectively. Unlike modular approaches where each component can be trained independently with smaller, task-specific datasets, end-to-end models often require extensive datasets that capture the full complexity of the task and environment. This need for vast amounts of data can be a bottleneck, particularly in robotics, where data collection is time-consuming and expensive. Furthermore, this leads to much bigger models, the training of which can have high hardware requirements. [12]

2.2.2 Foundation Models

Foundation models are a significant advancement in machine learning, representing large-scale, pre-trained models designed to serve as versatile platforms for a wide range of downstream tasks. Unlike traditional models that are trained for specific applications, foundation models are pre-trained on extensive, diverse datasets, enabling them to generalize across various tasks and environments. This concept, which has gained prominence in fields like natural language processing (NLP) with models such as GPT-3, is now making its way into robotics. [12]

In robotics, foundation models enable the development of systems that can quickly adapt to new tasks by leveraging the extensive knowledge acquired during pre-training. This adaptability reduces the need for extensive task-specific data collection and training, making it easier to deploy robots in varied and dynamic environments. The pre-training step itself requires large datasets, which are much less available in robotics compared to fields like NLP, and is computationally intensive. Despite these challenges, foundation models promise the ability to be fine-tuned with relatively little data and lower computational resources. [11, 13]

2.2.3 Imitation Learning

Imitation learning is a technique in machine learning where a model is trained to mimic the actions of an expert by learning from demonstration data. In the context of robotics, this typically involves collecting a dataset of demonstrations where a human operator manually controls the robot to perform the desired task.

In contrast to imitation learning, reinforcement learning (RL) is another widely used strategy in robotics. RL involves an agent that learns to make decisions through trial and error, receiving rewards for actions that move it closer to a specified goal and penalties for actions that do not. Over time, the agent optimizes its behavior to maximize cumulative rewards.

However, this trial-and-error approach often requires numerous iterations, which can be challenging to execute safely and efficiently in real-world robotic systems. Additionally, designing a reward mechanism that accurately evaluates the outcome of each trial can be difficult, especially in complex or nuanced scenarios.

Imitation learning, on the other hand, directly leverages expert knowledge, making it a more practical choice for tasks that are difficult to formalize with a reward structure. Its effectiveness, however, relies heavily on the quality and diversity of the demonstration data, as the model's ability to generalize to new situations depends on the breadth of experiences captured during training. [7]

2.3 RT-1

The RT-1 model [2, 14] was presented as a joint effort between Robotics at Google, Everyday Robots, and Google Research, at the end of 2022. The motivation is the following: End-to-end robotic learning generally relies on task-specific datasets that are narrowly tailored towards the robots intended tasks. This is similar to the traditional supervised learning approach in fields like computer vision and NLP,

where data is collected, labeled, and then utilized to train a model for specific tasks, with minimal interaction between the different tasks. In recent years however, there has been a shift in those areas, moving away from isolated, small-scale models and datasets towards large, general models that are pre-trained on extensive, diverse datasets. These models are able to absorb experience from large datasets to learn general patterns across tasks, allowing highly improved generalisation to unseen tasks compared to the traditional approach. While removing the need for task specific datasets is generally appealing in many domains, it is of very high significance in robotics, where the collection of datasets is typically very costly due to the need of either engineering-heavy autonomous operation or expensive human demonstrations. The purpose of RT-1 is therefore to investigate if it is possible to train a single, capable, multi-task model on data consisting of a wide variety of robotic tasks, and to find out if such a model brings the same benefits observed in other domains, namely zero-shot generalization to new tasks, environments, and objects.

With RT-1, the authors present a model architecture along with a significant training dataset that fulfills those requirements, as well as demonstrate the success of this model. The basic function of RT-1 can be observed in Figure 2.1: The model takes a natural language instruction, along with a history of six RGB images, as input, and returns a eleven-dimensional action description as output: seven dimensions for arm movement, three dimensions for base movement, and one dimension for terminating an episode. Note that the three dimensions for base movement are irrelevant for this thesis, as a stationary robot is used, it is only dealt with the seven-dimensional arm movement vector (and the termination signal) from here onwards.

The following section describes how this is done, including a summary for each of the used components, a description of how they interact, and an overview of the results the authors achieved with RT-1.

2.3.1 Foundations

The architecture of RT-1 involves several existing components, which are introduced here.

Transformers

Transformers are a type of deep learning model architecture that has significantly impacted various fields of artificial intelligence since their introduction in 2017 [15]. Transformers are built around the self-attention mechanism, which allows them to analyze and prioritize different parts of input data relative to one another, regardless of sequence order. This makes them particularly powerful for capturing complex relationships and patterns within data. While they initially revolutionized natural language processing tasks such as translation and text generation, enabling recent break throughs like e.g. ChatGPT, their versatility extends far beyond language. Transformers have also been successfully applied in areas like computer vision , where they enhance image recognition and generation [16], and in time-series forecasting , where they improve predictions by modeling dependencies across time steps [17].

Transformers have also been proposed for application in robotics, and have shown great success especially in approaches using language instructions for task specification [18–20]. RT-1 goes one step further than previous approaches, by treating the mapping of language and vision observations to robot actions as a sequence modelling problem, and using a Transformer to learn this mapping.

EfficientNet

EfficientNet is a family of Convolutional Neural Networks (CNNs) introduced in 2019. The key innovation of EfficientNet is a method called compound scaling: CNNs are often developed at a fixed resource budget, and eventually scaled up for better accuracy, once more resources are given. Traditional methods scale one network dimension at a time and independently of the others (depth, width or resolution). EfficientNet's compound scaling scales all three dimensions uniformly, using a simple yet highly effective compound component. Based on this, the authors develop a family of models (EfficientNet-B0 (5.3M params) to B7 (66M params)), which achieve much better accuracy and efficiency than any previous CNNs. In RT-1, the B3 version of EfficientNet is used. It is pre-trained on ImageNet, a large-scale image dataset [22]. [21]

Universal Sentence Encoder

The Universal Sentence Encoder (USE) is a pre-trained model introduced in 2018, designed to convert sentences into high-dimensional embedding vectors. It is specifically targeted towards transfer learning

to other NLP tasks, meaning that its goal is to provide a general-purpose embedding that can be used in a wide range of NLP applications. It produces fixed-length embeddings of 512-dimensional vectors, regardless of the input sentence length. The embeddings can be used to determine the semantic similarity between short pieces of text. [23]

FiLM

Feature-wise Linear Manipulation (FiLM) is a conditioning method for neural networks introduced in 2018. It works by introducing additional layers to a CNN. FiLM layers carry out simple, feature-wise affine transformations on a network's intermediate features, conditioned on an arbitrary input. This significally alters the CNN's behavior depending on the conditioning input, allowing the overall model to carry out a variety of conditioning tasks. In the case of RT-1, it is used to extract task-relevant image features based on the language instruction. [24]

TokenLearner

Presented in 2021, TokenLearner is a neural network module designed to improve the efficiency of vision Transformers by dynamically generating a small subset of tokens from the input.

In the case of vision Transformer, each input image needs to be split up into smaller parts that can be processed by the Transformer, called tokens. Traditionally, images are tokenized by dividing them into thousands of parts of equal importance, which can make Transformers intractable for larger images or videos. TokenLearner is a learnable module that takes images as input, determines which parts of the image are "worth processing", and based on that generates a small set of tokens. In RT-1, it works by selecting relevant image tokens based on their information, and passing only important token combinations on to the subsequent Transformer layers [2]. By doing so, it saves memory and computation by more than half, without impacting classification performance. [25] While these tokens can technically be extracted and interpreted to further understand which parts of the image are passed to the Transformer layers, this is complex in practice due to the end-to-end design of RT-1

2.3.2 Method

One of the main contributions of RT-1 is its network architecture. It is designed to help the model achieve its goals of being efficient and generalizable. This section gives an overview of its architecture, the reason why the different components where chosen and the way they work together.

Two aspects are of high importance for effective robotic multi-task learning: On one hand, a high-capacity model is needed. For this reason, a Transformer model was chosen as the central part of the architecture: These types of models excel particularly when the goal is to learn many tasks conditioned, like in this case, on a language instruction. The second aspect is efficiency: To effectively control a robot, a model must be able to run inference in real time, presenting a major challenge for Transformers. This is addressed by encoding the high-dimensional inputs and outputs (language instruction, camera images, robotic actions) into compact token representations to be used by the Transformer. In the following, the different parts of the architecture are discussed in detail. Figure 2.1 shows the model inputs, outputs, and a simplified version of the architecture.

Input Tokenization

The model takes two input types: A history of six RGB camera images on one hand, and a natural language instruction on the other hand. The images are tokenized by passing them through an EfficientNet-B3, pre-trained on ImageNet, which takes six images of resolution 300x300 as input and returns a spatial feature map of shape 9x9x512 per image. This feature map is then flattened into 81 visual tokens, which can be passed to the later layers of the network.

To include the natural language instruction, it is first embedded using the universal sentence encoder, which transforms it to a vector of length 512. This vector is then used as an input to identity-initialized FiLM layers, which are added to the pre-trained EfficientNet to condition the image encoder, to extract task-relevant image features early on. Inserting FiLM layers into the interior of a pre-trained network would normally disrupt the intermediate activations and thereby negate the benefits of using the weights obtained by pre-training the EfficientNet on ImageNet. To avoid this, the weights of the FiLM's dense layers, which produce the FiLM affine transformation, are initialized to zero, which allows the FiLM layers to intitally act as identities, preserving the function of the pre-trained weights.

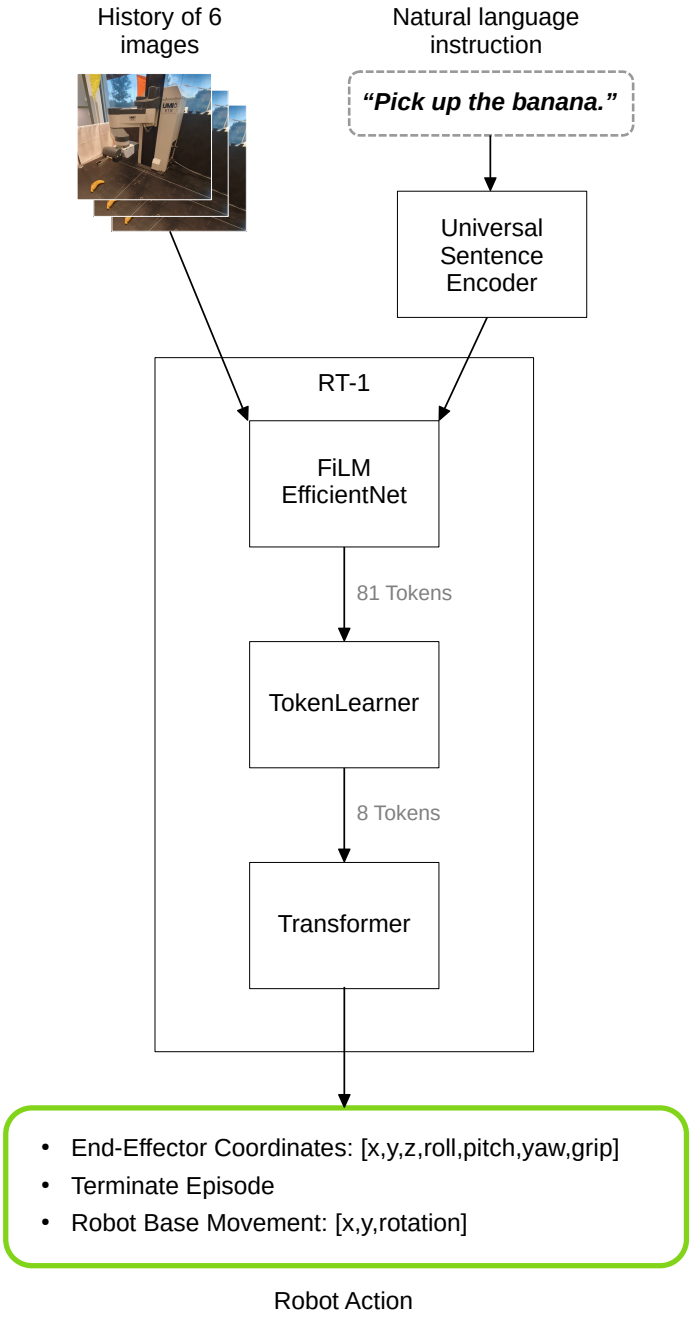

Figure 2.1: A simplified version of the RT-1 architecture, showing the input of an image history and language instruction, application of the various modules, and the robot action output format.

The output after the image and instruction tokenization is 81 vision-language tokens per image. To further speed up inference, the number of tokens that the Transformer needs to attend over is further compressed using TokenLearner. From the 81 tokens that come out of the FiLM-conditioned EfficientNet, TokenLearner derives only 8 final tokens.

Transformer

These eight tokens are then concatenated with the tokens of the other images in the history and the positional encoding required for the Transformer is added, forming 48 total tokens with are finally passed on to the Transformer layers. A decoder-only sequence model with 8 self-attention layers and 19M total parameters is used, outputting action tokens. To calculate loss, a standard categorical cross-entropy entropy objective and causal masking are used, which has been proven successful in previous Transformer-based controllers [26, 27].

Action Tokenization

RT-1 considers 11 dimensions of robotic action: seven variables for arm movement (x, y, z, roll, pitch, yaw, gripper opening), three variables for base movement (x, y, yaw), and a discrete variable to switch between the modes base movement, arm movement, and terminate episode. The continuous action space of every variable us discretized into 256 bins, for efficient use in the Transformer. The bins are uniformly distributed within the bounds of each variable.

2.3.3 Training Data

The main goals while gathering training data was to make the model able to generalize to new tasks, backgrounds and to handle various distractors. To achieve this, a large, diverse dataset of robot trajectories was collected, consisting of around 130k robot demonstrations including different tasks, objects and environments. Mobile manipulators from EverydayRobots are used as the robot embodiment. The data collection was conducted in a series of office kitchens, using teleoperation with two virtual reality remotes. The tasks include picking, placing, opening and closing drawers, getting items in and out of drawers, placing elongated items up-right, knocking them over, pulling napkins and opening jars.

2.3.4 Performance

The authors test RT-1 on multiple aspects, against two baseline state-of-the-art architectures: Gato [26] and BC-Z (as well as an extended version of BC-Z, called BC-Z XL) [28]. For the tests, both of these models are trained on the data collected for RT-1, so the comparison only considers the performance of the actual model architecture, and not the dataset, giving the baseline models an advantage, as the RT-1 dataset is assumed to be better than the individual ones.

The models are compared on the performance on seen tasks, and the generalization capabilities to new tasks, distractors and backgrounds. It is found that RT-1 outperforms both Gato and BC-Z in all aspects: On seen tasks as well as on novel tasks, RT-1 performs around 25% better than the next-best baseline. On distractors and changed backgrounds, it performs 36% and 18% better than the next baselines, demonstrating impressive degrees of generalization and robustness of RT-1. The performance in a realistic kitchen scenario is then tested, where novel tasks, distractors, and backgrounds come together. RT-1 once again outperforms the baseline models.

The authors then investigate the possibility of improving RT-1 by incorporating data from heterogenous data sources. To do so, they test two scenarios: Incorporating training data from a simulated version of the Everyday Robots manipulator, and incorporating data from an entirely different robot embodiment. For the simulation data, the authors find that there is a significant increase in performance on tasks and objects that are only seen in the simulation data, while there is no performance loss at all compared to only using the real training data. For testing incorporation of data collected on another robot, a KUKA IIWA is used. This robot is different from the Everyday Robots manipulator in appearance, action space, and environment, as well as the demonstrations being collected by an RL agent instead of a human. With the combined training data, a significant performance increase on tasks that are only in the KUKA dataset is observable, while original task performance in minimally reduced. Also, the authors show that RT-1 trained exclusively on KUKA data has a 0% success rate when evaluated on the same tasks on the Everyday Robots manipulator, confirming that it is difficult to transfer behavior

from another robot morphology. However, these tests indicate that RT-1 is able to acquire new skills through observing other robot's experiences.

Finally, the authors evaluate the importance of dataset diversity vs dataset size, as this plays an important role in the field of robotic learning, where data collection is particularly expensive. To do so, they first remove the number of examples per task, to reduce dataset size. A general trend of performance decrease is observable, as well as a steeper trend of decreasing generalization. To reduce data diversity, it is simply the number of tasks that is reduced. With that, the decrease of performance as well as generalization is much steeper. This shows that data diversity is more essential than data quantity.

2.4 Open X-Embodiment and RT-1-X

Recent advancements in machine learning have shown that large-scale training on diverse datasets can lead to general-purpose models that even outperform their narrowly targeted counterparts, trained on smaller, task specific data, by leveraging the benefits of positive transfer between domains. Increasingly, the go-to approach to tackle a given narrow task e.g. in vision or NLP, is to adapt a general-purpose model. In the previous section, findings from RT-1 were presented, which show that this approach could also work in robotics. In this domain however it is hard to apply on a larger scale, since as previously discussed, datasets for robotic interaction are hard to come by. Even the largest existing robotic datasets are a fraction of the size of their vision or NLP counterparts, in addition robotic datasets are very often still narrow along some axes of variation, either all collected on a single environment, or only demonstrating a small set of tasks and objects.

2.4.1 Method

Open X-Embodiment addresses this issue by gathering a number of robotic datasets collected at many different labs, on different robots and in different environments, and assembling them into one unified dataset. While each individual dataset might be too narrow to train general purpose policies, the union of many such datasets can drastically improve coverage. The authors also provide open-source tools to facilitate further contributions to Open X-Embodiment, aiming to get closer to the dataset scale and diversity of other domains in the future. To evaluate the resulting dataset as well as the potential of positive transfer in robotics, the authors train several state-of-the-art models with the Open X-Embodiment data and compare the performance to the baseline of training only on task-specific data.

Dataset Composition

At the time of publishing, the Open X-Embodiment dataset consisted of over 1M real robot trajectories, collected from 22 different robot embodiments. It was constructed by gathering 60 existing robot datasets from 34 robotic research institutions around the world. Since then, more datasets have already been added. All datasets are converted into a consistent format: The RLDS (Reinforcement Learning Datasets) format was used, which accommodates the various action spaces and input modalities of different robot setups. This format also allows for efficient data handling in all major deep learning frameworks.

Figure 2.2 shows a breakdown of the datasets, scenes and trajectories by robot embodiment, dataset skills, and dataset objects. While not being equally distributed, it shows a great diversity along all axes.

RT-1-X

To evaluate the performance benefits of Open X-Embodiment training, models with enough capacity to productively make use of such large and heterogeneous datasets are necessary. The authors use the previously discussed RT-1, and its derivative RT-2. In this section, the main focus lies on the model based on RT-1, as this is the one used in the course of this thesis. RT-2 is further discussed in Section 8.3.1.

The architecture of RT-1 has already been discussed in the previous section. Compared to the original implementation, the authors of Open X-Embodiment use a history of 15 images as model inputs, instead of the original 6. There is no reason given for this change. However, other researchers working on the RT-1 model after its release have also used a history of 15 images, stating that it helps accommodate longer episode length.

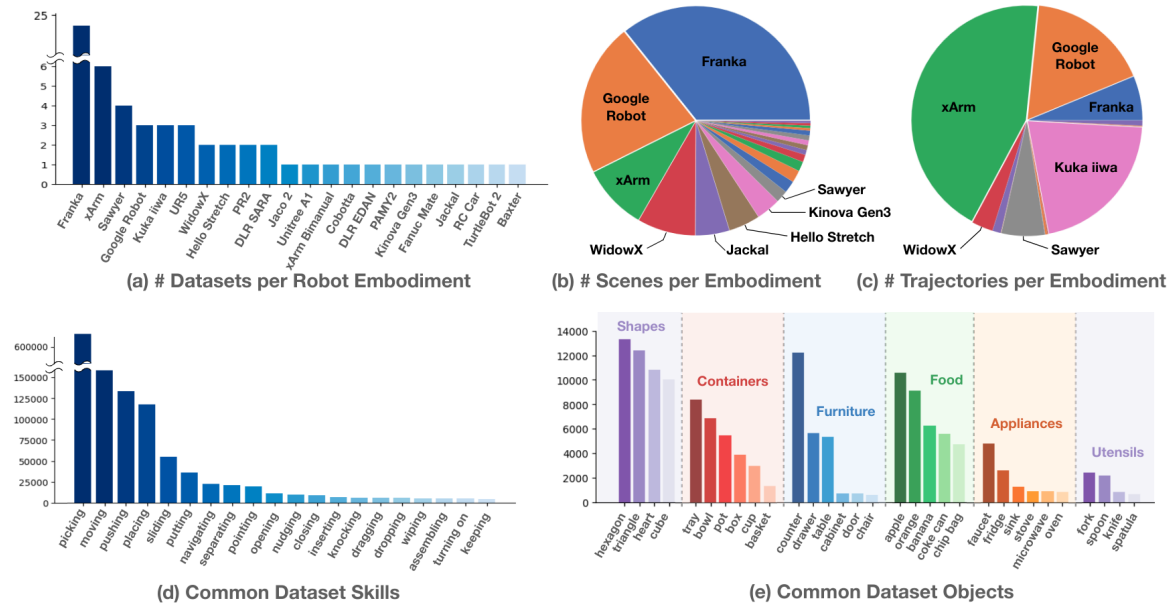

Figure 2.2: The Open X-Embodiment dataset features a large variety of embodiments, skills and objects. The composition of the datasets is shown here. [1]

One challenge in training models on the heterogeneous X-Embodiment dataset is the significant variation in observation and action spaces across robots. To address this, a coarsely aligned action- and observation space is used across datasets: From each dataset, one camera view is selected, resized to a common resolution, and used as model input. Camera observations naturally vary substantially across datasets, as they are using different camera positions, scenes and embodiments. The dataset's action set is converted to a 7 DoF end-effector action and normalized, so the model's output can be interpreted differently depending on the embodiment. The action space is however not aligned across datasets, meaning action values can either represent relative positions, absolute positions, or even velocities, according to the original control scheme of the robot. The same output actions can therefore induce very different motions for different robots.

2.4.2 Performance

The authors want to test the models trained on their X-Embodiment data in three aspects: Positive transfer between robots, generalization to new tasks, and the influence of different design dimensions. It should be noted that for the evaluation of their X-Embodiment trained models, the authors use a data mixture from only 9 different embodiments, as this was all available data at the time of the experiments.

Firstly, it is evaluated if policies can benefit from positive transfer, meaning that co-training on data from different robots improves performance on the training task. The authors find that in domains that have only small-scale domain specific datasets, co-training with X-Embodiment data leads to significant performance improvements, showing that these domains benefit substantially from positive transfer. On domains where there are already large-scale datasets available, RT-1-X does not show significant improvement over the performance on RT-1 trained only on the domain specific data.

The evaluation of generalization capabilities and influence of different design decisions is done only using the RT-2-X model. It is therefore unclear how the findings translate to RT-1-X. However, the experiments show that generalization to new objects, backgrounds and environments is not influenced by the addition of the X-Embodiment data. On the other hand, it is shown that the model running on one robot is able to execute skills that were only present in the dataset of another embodiment, demonstrating that incorporating the data from other robots into the training improves the range of tasks of a robot. When experimenting with different design dimensions, the main findings are that using a history of images as input instead of a single image significantly improves generalization performance, and that a higher model capacity enables a higher degree of transfer across robotic datasets.

2.5 UMI RTX Robotic Embodiment

The robot used in the course of this thesis is the RTX model from Universal Machine Intelligence Ltd. (UMI). This section will give an overview of the technical aspects of the robot, and also give insights into the history and previous use cases. It should be noted that outside this section the UMI RTX robot will be referred to as the UMI robot or just the UMI, in order to avoid confusion with the RT-X models (RT-1-X and RT-2-X) from Open X-Embodiment.

2.5.1 History and Use Cases

The RT series is a family of robots that was first introduced in the mid 1980's, by the British company Universal Machine Intelligence Ltd., under the technical direction of Tim Jones. The goal was to develop a robot that sits between the two types of robots that were available on the market at that time: On one hand the cheap but incapable toy robots, on the other hand powerful, but highly expensive industrial machines [29]. The original plan was to develop a mobile unit called the R-Theta, which was developed in 1983 [30]. It was soon realized that there wasn't an immediate market for a mobile personal robot, which is why UMI instead released a low-cost, stationary version of their arm, called the RTX. According to Jones, this was the first "co-bot" in existence [31], as it was capable but still safe to use around humans due to its low-powered motors and belt drives, which were meant to act as mechanical fuses [32]. UMI, which was later acquired by Oxford Intelligent Machines (OxIM) and even later by the French Afma Robotics, went on to develop further iterations of the RTX: The RT100 was similar in design but more robust, targeted towards light industrial applications, with a higher reach and payload capacity and the RT100+, its more powerful update. With the RT200, the working envelope was greatly extended with the addition of a new axis: The entire assembly was placed on a linear rail system, available in lengths up to 6m. It was additionally improved in terms of speed and accuracy. [33, 34]

Right after its introduction, the RTX was widely adapted in research settings, with multiple institutions using it for teaching [35] and experimentation with different technologies [32, 36–38]. The RTX was especially popular in the domain of rehabilitation and healthcare robotics: A survey from 1990 shows that the UMI RTX was the most widely used robot worldwide in the area of rehabilitation robotics at the time [39]. This was due to its unique market position inbetween educational and industrial robots [39], and its wide range of movement and safety features [40]. Among other use cases, the RTX and its successors were used to support patients with tetraplegia in France [41] and as a voice-controlled workstation for disabled programmers by the Boeing company in Seattle [42, 43], as well as in various other research projects related to rehabilitation [44–46]. More applications of the UMI robots can be found in the dissertation of Hillman from the year 1992 [47]. A custom robot, similar in architecture to the RT models, was built by UMI to efficiently feed Icelandic cod from a conveyor belt to a fish beheading machine [33, 48].

After the 2000's, not much work was done with the UMI RT robots. It is unclear how many of them are still in working condition. At KU Leuven, two master's theses were done in the years 2015 [49] and 2016 [50], in which the authors worked with a UMI RTX robot that seems to be or have been in working order at KU Leuven. Since 2022, there have been efforts to "revive" the UMI RTX at the University of Amsterdam Intelligent Robotics Lab, which was first used in 1992 to play chess [51], and later as a plotter [52]. Control code that runs on modern Linux distributions was written by Visser [53]. This was extended by Garde and Massa, who built a ROS2 interface around the control code, as well as a graphical user interface to easily control the robot and some computer vision capabilities [54, 55].

2.5.2 Technology

The UMI RTX is a robot of the SCARA (Selective Compliance Assembly Robot Arm) type with the addition of vertical travel. It has 7 degrees of freedom: The rotation of the elbow and shoulder joints, yaw, pitch, and roll of the wrist (end effector), the height of the entire arm assembly (z axis), and the opening of the gripper. Figure 2.3 shows the UMI robot at the Intelligent Robotics Lab, which will be used for this thesis. Due to its SCARA layout, the range of motion of the robot is kidney-shaped. A specific design choice is that the elbow joint and the wrist's yaw joint are driven through a combined spindle, meaning the wrist yaw always stays the same in relation to the robot base when moving the elbow joint. With certain yaw orientations, this leads to the wrist hitting its end stops when moving the arm in certain directions, causing the belts to skip. The official manual acknowledges this and states that such movements should be avoided, unfortunately without giving further information about the issue. [56]

Figure 2.3: The UMI robot used in this thesis, in its setting at the Intelligent Robotics Lab of the University of Amsterdam.

The control code used in the course of this project is the ROS2 interface by Garde and Massa, Garde and Massa, who use the code by Visser for low level controls and serial communication to the robot [53]. There were however some issues with parts of this code which were fixed in the course of this project. This will be discussed in later parts of this thesis.

Chapter 3

System Integration

To conduct experiments with the RT-1-X model and the UMI robot, it is first necessary to set up all required components, and build the necessary interfaces between them. There are three main areas that require consideration:

- Robotic control: All components that are responsible for receiving robotic commands and reliably and accurately execute them on the UMI robot. This includes software as well as hardware.
- Model inference: The RT-1-X model must be able to run inference with pictures from the environment camera, and output actions that are suitable for the UMI.
- Infrastructure: Robotic control and model inference need to be brought together in a way that allows for easy debugging, changing of variables for experimentation, and ideally, the ability to adapt to other robots and models.

Most components for robotic control and model inference are already available in a basic form. However, there are many adaptations required to reliably make them work for this use case. This chapter explains the process of getting the individual components ready and finally connecting them. A special emphasis is placed on the verification of all components, to mitigate any errors that could distort the results of later evaluations. Most of the components presented in this chapter are also made available for reproduction and further research.

3.1 Robotic Control

To enable the robot to be controlled by the RT-1-X model, a control interface needs to be established. Even though the UMI RTX is a rather old robot that has never been in widespread use, it has been well maintained in the Intelligent Robotics Lab at the University of Amsterdam and there is modern code for it available. A driver was written by Visser in 2022[1], and a ROS2 interface for it was developed in 2023 in the course of an internship by Garde and Massa [54]. While these components provide a good starting point for this project, on closer inspection it became apparent that there were some issues with them, especially the ROS2 interface:

- The ROS2 interface was bundled together with a computer vision, simulation, and GUI component, which added much overhead and unnecessary dependencies. Also, it did not support Ubuntu 22.
- There were issues in the control logic of the ROS2 interface, including race conditions.
- The inverse kinematics component in the ROS2 interface code did not use the entire robot workspace, as well as leading to glitches when using values close to the movement limits. This often led to collisions.
- The Z Axis limit was too high, meaning the robot could not reach all the way down to the workspace, making it impossible to pick up small objects.

To address these issues, a fork of the existing ROS2 interface repository was created [6], with the goal of turning it into a minimal, correct ROS2 interface for the UMI robot. This section discusses how the various problems were addressed, to allow for further development of this project without limitations from the robot layer.

[1] `https://github.com/physar/umi-rtx`

3.1.1 Removing unneeded dependencies

As it is, the LAB42_RTX_control bundles multiple functionalities, which is also described in its project report [54]. The main ones are:

- Arm control: The code necessary to move the actual robot, including inverse kinematics and communication with the hardware
- Computer vision: Robot is able to detect a banana in the workspace and pick it up
- ROS2 communication: Receiving and sending messages with ROS2
- GUI: a simple graphical interface that allows controlling the robot

While the arm control and ROS2 communication components are essential for the robot to work, the computer vision functionality is not needed for a minimal ROS2 node. Additionally, this part of the code requires a specific version of CUDA to be installed which is not compatible with the version of TensorFlow used in RT-1-X, and it also uses the ZED camera, which leads to conflicts when using it in the RT-1-X code. In the forked repository, the computer vision code is removed, as well as its dependencies.

The GUI component is helpful for manually experimenting with the robot, although it constantly publishes ROS2 robot instructions, which makes it impossible to control the robot with other nodes while the GUI is running. To circumvent that, and enable the functionality of switching back to GUI controls at any time, an additional button called "ROS publishing" is added to the GUI. This button allows to temporarily deactivate the GUI controls, a second click of the button instantly reengages the GUI controls.

3.1.2 Bugs in control logic

The repository contained multiple bugs in the raw control logic of the robot, which prevented reliable control of the robot via the ROS2 interface. Most issues did not occur when controlling the robot via the GUI, indicating that the ROS2 interface was not tested with external control nodes. As the goal is to have a universal ROS2 control interface that can be controlled externally, those bugs need to be addressed.

3.1.3 Inverse Kinematics

Inverse kinematics describes the process of computing the motor commands required to bring the end effector to a certain position. This is not a trivial process, as for a single end effector coordinate, there can be one, multiple, or no solutions. The LAB42_RTX_control has a component which solves this inverse kinematics problem using Pinocchio, a popular rigid body dynamics library [57]. When experimenting with the robot using manual controls, it became apparent that there were issues coming from this inverse kinematics implementation.

The cause of this issue is that the entire UMI control code, including the inverse kinematics implementation, handles robot coordinates in the cartesian format. The UMI robot however has a kidney-shaped range of motion, which cannot be trivially described in cartesian coordinates. Figure 3.1 shows the UMI's range of motion.

It is also difficult to define the minimum and maximum values for one single coordinate, as this is dependent on the other coordinates. This is not handled in the UMI control code: When choosing coordinates that are outside the limits of movement of the UMI robot via the ROS2 interface or the GUI, the inverse kinematics lead to glitching, returning nonsensical motor commands, which often lead to very different end effector positions and even collisions.

Also, when providing the robot arm with coordinates that are close to its own base, it occasionally fails to reach these coordinates, despite being physically capable of doing so. This problem however only occurs when the target is not reachable in a simple move from the start position, meaning the robot would have to change its position entirely to be able to reach its target. This problem is illustrated in Figure 3.2, based on a simulation of the UMI. For both pictures, the same target coordinates were given to the robot, shown by the yellow star symbol. The only difference is that on the left picture, the arm was in a start position where the elbow was bent to the right, making it possible to get to the coordinates easily. In the right image, the start position had the elbow bent to the left. To get to the desired coordinates, the arm would have to straighten out first, moving away from the target, before then bending to the right, to make it able to reach the target. The inverse kinematics does not do this, and instead settles on a position that is somewhat close to the target, as shown in the picture.

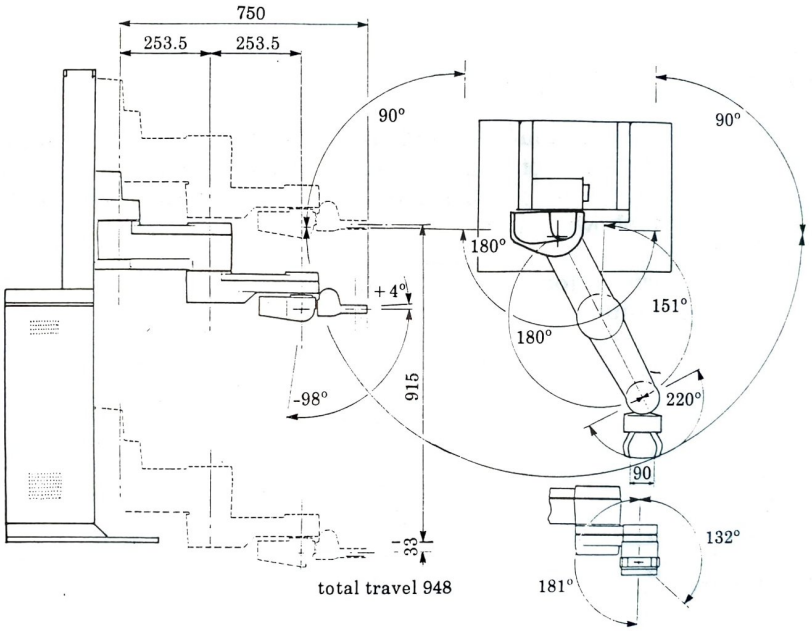

Figure 3.1: The UMI robot features a kidney-shaped range of motion, which makes kinematics in cartesian coordinates a challenge. [58]

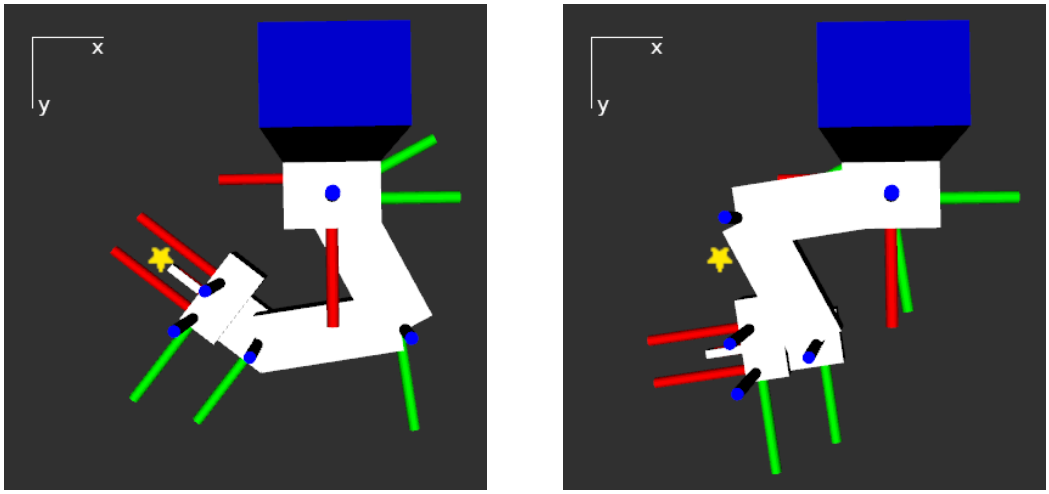

Figure 3.2: Despite identical target coordinates (indicated by the yellow star), the UMI takes on different positions depending on its starting position, and fails to reach the correct pose when starting from a disadvantageous position (right).

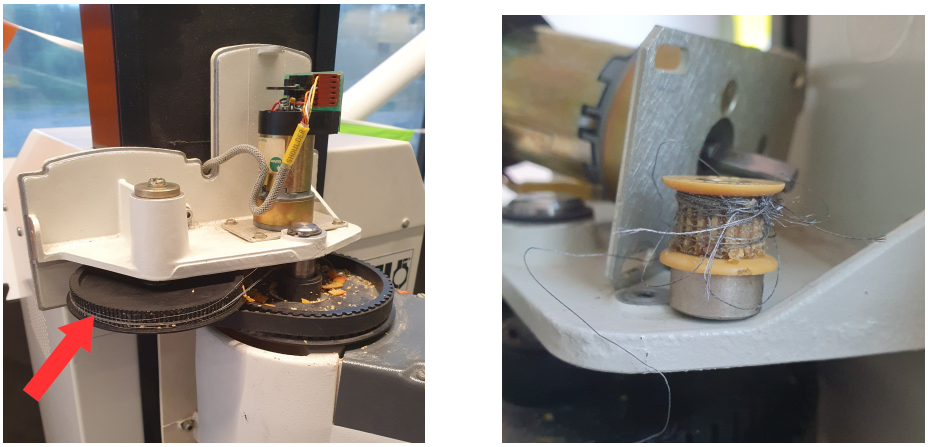

Figure 3.3: Certain parts of the UMI robot wear down fast due to its age. Left: Shoulder assembly with the broken belt highlighted with the arrow. Right: Broken pulley, destroyed by the remaining metal parts of the belt.

Further, certain combinations of movements output by the inverse kinematics lead to the control belts slipping, which leads to imprecise following movements as well as wearing down the hardware.

It was tried to find a solution to those issues, to improve or replace the inverse kinematics node, but it was ultimately decided that this is outside the scope of this thesis. To avoid problems stemming from this in the course of this project, the coordinate space will be reduced to a space that was found to work without any problems through experimentation. The minimal and maximal coordinate values that were determined though experimentation can be found in Appendix A.1.

3.1.4 Hardware Issues

The UMI robot uses timing belts that connect the motors to the respective joints. After around three months of continuous experimentation, the belt on the shoulder joint wore out, making it unusable. The metal reinforcement strands in the belt also destroyed the associated pulley. Figure 3.3 shows the two broken parts.

Enough replacement pulleys were available in the Lab, and the pulley could quickly be replaced. Two spare belts in the correct size were available in the Lab, however after replacing the first belt, the other two only held up for several days. This is presumably due to them not being moved and used in over 40 years, making the rubber loose flexibility and become brittle. To continue this project, it was therefore necessary to find suitable replacement parts.

As the UMI company is no longer existent, getting original spare parts was out of the question. The maintenance manual did not give any information on the belt specification, apart from an internal part number, the belts themselves had no information written on them as well. The old belts were measured, resulting in a measured width of 6mm, a tooth count of 175, and a pitch (distance between two teeth) of 2mm. Research on available standard parts yielded two results: The GT2 standard, a belt widely used in consumer 3D printers, and the MXL standard, a more specialized type, both fitting the specification. The only difference between the two types is the GT2 has a pitch of 2mm, whereas MXL has a pitch of 2.03mm, which made it impossible to reliably determine by measuring.

Both of these belt types are not commonly available in a belt length of 175mm, the only available products had a shipping duration of multiple weeks, which was close to the deadline of this project. In an effort to repair the robot faster, open-ended belts of both types were ordered, which were available within a few days. When testing them manually on the pulley, it was found that the MXL type belt was the correct one. It was then tried to glue the belts to the correct size, which unfortunately only worked for a few tries before the glue point broke. Finally, the problem could be solved with the 175mm belts, as soon as they arrived.

3.2 Model Inference

The authors of Open X-Embodiment provide a minimal example for running inference with the RT-1-X model from a pre-trained checkpoint. This example however does not consider input and output processing: Arbitrary arrays are used as model inputs instead of real images and language instructions, model outputs are not scaled to a robot action space, and only one inference step is run. The following section describes the steps that were necessary to process inputs from the UMI environment to run inference correctly. To assess the correctness of the inference pipeline before running inference with the UMI robot and to observe the effects of different factors, inference is run with the images and instructions from an existing dataset from Open X-Embodiment.

Notably, two different implementations of the RT-1-X model, along with their respective checkpoints, are available in the official repository: A version based on Tensorflow, which was initially published along with the Open X-Embodiment paper, as well as a version based on the Jax framework that was published months after the initial release. Neither the paper nor the repository description clearly outlines the differences between the two versions. During initial experimentation, it was found that both versions deliver similar output. Since training examples are only available in a Jax based implementation, the work in this thesis is based on this version of the RT-1-X model implementation.

3.2.1 Model Inputs

The RT-1(-X) model takes two different kinds of input: Images from the environment, and natural language instructions. The processing of these inputs before inference is crucial, as it was found that even slight differences in input processing leads to meaningless inference output. Unfortunately, the inference example does not show how this processing is done, and there is no further documentation available. The correct processing steps were determined by studying the model code, as well as consulting other developers working with RT-1-X via GitHub[2][3]. This subsection details how images are processed and used in the inference code.

Image Processing

Correct image processing is essential before running inference. It has been found that when not processed correctly, even slightly misformatted images have big negative impact on model performance. While neither the RT-1-X inference code, nor the according paper give information about the exact required image format, it can be reconstructed by observing how processing is done in the training code.

Firstly, a picture is taken with the ZED camera module, which returns a four channel image in the BGRA format. This is converted into the required RGB image array by first converting it to three channel RGB with OpenCV, and then converting it into a numpy array. Two transformations then need to be applied with the TensorFlow image module: First, the image is resized with padding to a size of 320x256, the result is then resized again without a pad to 300x300. While the RT-1 model requires an image size of 300x300, it is unclear why the first resizing is applied. However, as this is how the images are processed in training, the same needs to be applied for inference. For completeness, it was tried to run inference without this additional scaling, which negatively impacted performance.

History of Observations

RT-1-X uses a history of the last 15 observations for each step of inference. This is not considered in the inference example, as each inference step is run with random data. While it is relatively clear how to assemble the observation history from the shape of the example inputs, it is unclear how the history should be handled at steps 1-14 of inference, where a full history of 15 images is not yet available. The options are either to instantiate the image history with 15 images of the initial workspace with the robot in its initial position, or simply to initialize the history as an array of zeroes. Both variants were tried, no difference in performance could be found between them.

3.2.2 Model Outputs

The RT-1-X model outputs an 8-dimensional action vector, which is then used to control the robot. The Open X-Embodiment datasets that RT-1-X is trained on are however not aligned in their action spaces:

[2] https://github.com/google-deepmind/open_x_embodiment/issues/61
[3] https://github.com/google-deepmind/open_x_embodiment/issues/57

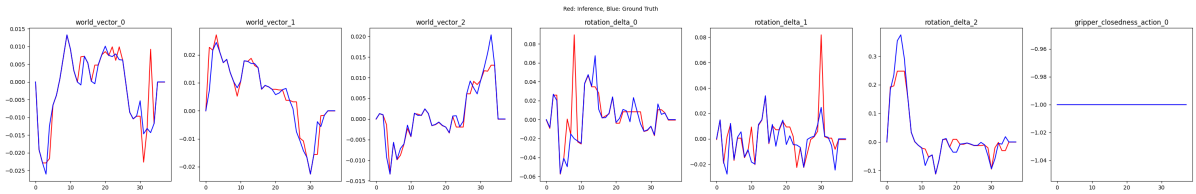

Figure 3.4: The RT-1-X model is run on images from its pre-training dataset, to validate the inference pipeline. Output of RT-1-X run with images from BridgeData [59] is shown in red, compared actions from the dataset (ground truth) in blue.

their action values describe either absolute or relative positions or velocities, as well as each dataset using different coordinate ranges. This poses issues when using the model for a robot it has not been trained on. There is no certainty over which interpretation or scaling should be used.

Action Interpretation

To account for different action interpretations, both absolute and relative position control will be considered in the zero-shot evaluation. Velocity control is not considered, since only a small amount of datasets use this action interpretation, and controlling the UMI by velocities would require new control logic.

Action Scaling

Before training, each datasets actions are scaled to a unified range, which is (-2, 2) for positional values, and $(-\frac{\pi}{2}, \frac{\pi}{2})$ for rotational values. To use the model output actions as absolute positions for the UMI, the values need to be scaled from this coordinate range to the UMI coordinate range. Scaling the values for controlling relative positions is less clear, as there is no constant maximum for a movement. It is assumed that the maximum movement at every step is half the action space for the respective axis. More information on the different movement ranges can be found in Appendix A.1

3.2.3 Validation

Before connecting the model to the UMI robot and evaluating the performance of RT-1-X on a new embodiment, it is crucial to verify the quality of the inference pipeline. There are multiple factors that can lead to incorrect inference output, such as misformatted inputs, incorrect processing, or bugs in the inference code, all of which can lead to poor evaluation results that are not representative of the model's actual performance. To ensure that the inference pipeline is working correctly, the model is run on a dataset of images and instructions from Open X-Embodiment, and the output is compared to the expected output. The dataset used for this validation is part of the model's training data, which should ensure that the model performs well on it. The same mode of validating the inference pipeline was used in the original Open X-Embodiment code[4].

The test is run with the Bridge dataset [59], one of the biggest datasets in Open X-Embodiment. The inference output compared with the ground truth (actions from the demonstration) is shown in Figure 3.4. It shows that the inference pipeline is working as expected, and is ready to start experimentation.

3.3 Infrastructure

The final step in the system integration process involves seamlessly integrating the robotic control and model inference components in a manner that supports easy debugging, experimentation, and future adaptability to other robots and models. This was achieved by developing a ROS2 node [4] that interfaces with the UMI environment, taking in images and instructions, performing inference using the RT-1-X model, and then sending the appropriate commands to the UMI robot.

To ensure flexibility and maintainability, the node was designed with a clear separation between the inference logic and the robot control code. This decoupling allows the inference code to be easily replaced or updated with alternative implementations, which is particularly valuable for ongoing experimentation

[4] `https://colab.research.google.com/github/google-deepmind/open_x_embodiment/blob/main/colabs/Minimal_example_for_running_inference_using_RT_1_X_TF_using_tensorflow_datasets.ipynb`

and potential future enhancements. The use of standard ROS messages for communication further enhances this flexibility, making the node readily adaptable to different robot types and configurations in future projects.

Chapter 4

Evaluating RT-1-X Zero-Shot on the New Embodiment

4.1 Introduction

After all required components have been set up and connected in the previous chapter, the performance of the RT-1-X model zero-shot on the unseen UMI embodiment will be investigated in this chapter. This directly addresses **RQ1**: *Can the RT-1-X model generalize to a completely unseen robotic embodiment, without any additional data, utilizing knowledge learned from the Open X-Embodiment dataset?*

Answering this question is a fundamental part in exploring the capabilities of RT-1-X on new embodiments. RT-1-X has already shown abilities of performing zero-shot in changed environments, e.g. with changed backgrounds and added distractors. Performance on completely unseen embodiments has not been investigated yet, even though a model that allows zero-shot generalization to new embodiments would be very meaningful in the landscape of robotic learning.

During initial experimentation, it became clear that the model failed to complete tasks meaningfully, highlighting potential limitations in its ability to adapt to novel physical embodiments. Recognizing these limitations, the focus shifted to exploring various environmental variables that might influence the model's performance. This included adjustments to camera positioning, task instructions, target objects, and background settings. The rationale behind this approach was to identify any factors that could potentially trigger signs of task completion or to understand better how different variables impact the model's behavior, to ultimately determine if the RT-1-X model can transfer any skills it learned from the Open X-Embodiment dataset, to the UMI environment.

This Chapter mainly explains the experimentation process. First, the experimentation environment is presented, it is discussed which properties are constant and which ones can change. Then, the process of experimentation is discussed, including how the model is used, how results are logged, and how they are evaluated. The result of this chapter is a set of 48 experiments, coming from a combination of these factors:

$$\textit{2 Tasks} \times \textit{2 Objects} \times \textit{3 Camera Positions} \times \textit{2 Action Interp.} \times \textit{2 Test Runs} = \textit{48 Experiments}$$

The individual factors will be further explained in the course of this chapter. The results of these experiments will be presented in Chapter 6, and their implications are further discussed in Chapter 7.

4.2 Experiment Design

This section goes into detail about how the experiments were designed to address the goals formulated in the previous section. First, it is discussed which factors are considered for investigation. Then, the method of conducting the experiments is explained, and finally, it is discussed how the results of the experiments will be assessed, and when an attempt will be considered successful.

4.2.1 Considered Variables

When trying to evaluate the effects different factors have on model performance, it is important to get an overview of all properties of the environment. They can be separated into factors that are constants

of the environment, factors that can be controllably varied, and factors that change, but cannot be controlled.

Constant Properties

Firstly, one single embodiment, the UMI RTX robot, will be used for all experiments. While the term "embodiment" technically only describes the physical robot itself, the terms "embodiment" and "environment" are closely related in this context, and cannot be completely separated in the course of experimentation. The "environment" that is observed by the camera consists of the embodiment itself, the workspace (background), target object, possible distractors, lighting conditions and any other elements that are visible on the observations. Using the UMI embodiment while keeping the rest of the environment constant to the pre-training data is not possible without having access to one of the training environments. Furthermore, the UMI is a large robot, permanently installed in a workspace at UvA's Intelligent Robotics Lab. In most real world applications, each robotic embodiment will have its own environment as well, differences in environment are part of the embodiment gap that needs to be bridged. The UMI robot permanently installed on a worktable, and this table has a fixed position in UvA's Intelligent Robotics Lab. This position and base workspace are taken as constant for this thesis.

Uncontrollable Properties

Environment properties that change, but cannot be controlled, are dangerous to the experiments as they could have an uncontrollable influence on the results. Only one factor of such category could be identified, and measures were taken to minimize its impact on experiment outcome.

The fixed position of the robot brings a challenge due to its position close to a window: The workspace lighting cannot be fully controlled. The window offers semi-transparent blinds, which diffuse the incoming light, preventing harsh shadows and uneven lighting, those blinds will be used for all experiments, as well as the room lighting turned on. Also, all experiments will be conducted between 10am and 5pm during the summer months, to get similar amounts of daylight each time. All of these measures lead the lighting being roughly similar in all scenarios. However, depending on the time of day as well as the weather, lighting can still vary slightly.

Controllable Properties

These are the environment properties that can be controlled during the experiments. It is their influence on model performance that should be evaluated by the experiments, they are therefore the most important factors. All of the identified controllable properties are described below.

Task and Target Object. The task, given as a natural language instruction, along with the target object, are arguably the most important factors of the setup, as they define what the robot is supposed to do in an experiment run. As the goal of these experiments is only to investigate generalization to new embodiments, it is essential to exclusively evaluate on tasks and objects that are present in the Open X-Embodiment training data. Simultaneously, the task must be reproducible with the resources available in the Intelligent Robotics Lab.

Pick-and-place tasks are very common in Open X-Embodiment as well as easily reproducible, which is why two tasks of this kind are chosen for evaluation: The first language instruction is simply "Pick up X", with X being replaced by the target object. The second evaluation task is slightly more complex, "Place X in the pan". As a toy kitchen environment is used in many of the training settings, two objects from this category are chosen as target objects: A toy banana and a coke can. These objects are among the most common objects in the entirety of Open X-Embodiment.

Workspace Setup. Besides the robot itself and the target object, the workspace that is observed by the camera consists of a background, as well as possible distractors. The basic setup used here is a black tabletop surrounded by a white curtain. As the UMI is fixed to this tabletop, changing the workspace is very limited. It is experimented with covering the tabletop with a white cloth, as well as placing more objects on the workspace as distractors.

Camera Position. A change to the camera position alters the model input significantly, as changing the position even by a few millimeters alters all pixels in the resulting image. Three different camera positions are therefore considered for evaluation, based on the camera positions of the training setups:

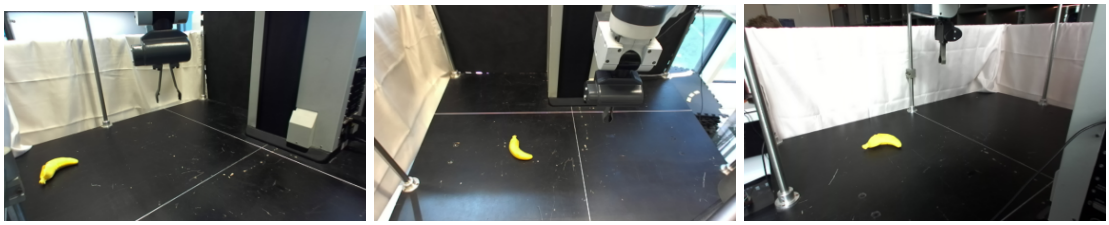

Figure 4.1: Multiple camera positions were evaluated, as changes in the camera perspective can change the entire perception of the environment. F.l.t.r: Camera position "Side", "Front", and "Shoulder".

- *Frontal*: Camera exactly opposite to the robot and looking down at an angle of 45 degrees.
- *Shoulder*: Camera looking "over the shoulder" of the robot: Placed to the side of the robot, looking down on the workspace.
- *Side*: Camera on the opposing side of the robot, but on the corner of the workspace, looking down.

Figure 4.1 shows sample images from all camera perspectives.

Action Interpretation. As the different training datasets come with different interpretations of action values, it cannot be clearly determined if the model outputs should be used as absolute coordinates or relative coordinates; this predicament has already been discussed in Chapter 3. To make sure that the model performance is not falsely rated as poor, experiments are conducted with both variations of action interpretation.

4.3 Experimentation Method

To ensure consistency and comparability across all experiments, a standardized method was established. This section outlines the key aspects of the experimental procedure, beginning with the technical considerations for running inference, followed by the approach to logging and classifying experiment results, and concluding with details on the sequence and structure of the experiments.

4.3.1 Technical Properties

Several variables of the inference process itself need to be set:

Inference Frequency. This is the frequency at which a picture is taken and a new action is predicted by the model. Due to the delay between commands and movements of the dated UMI robot, a control frequency of **0.2Hz (five seconds between actions)** is used, which is enough time for the robot to execute model actions before the next inference step is run.

Sample Size. To prevent random flukes to be rated as good performance, it is crucial to run each experiment more than once. However, as each experimentation run takes a significant amount of time (e.g. one run of 50 steps at 0.2Hz takes over 4 minutes, excluding preparation and analysis), increasing the sample size drastically increases the duration of experimentation. Therefore, each experiment is **run twice**.

Result Logging. The main observation will be by physically watching the robot during the experiment runs. However, to get a deeper understanding of the movements it executes and to be able to reliably compare different runs, the model outputs will be logged in three different ways:

- A plot of all output dimensions, including a picture of the environment and the language instruction, for an overview of the entire experiment and its outcome
- A heatmap visualizing the X and Y position of the robot after every inference step, to check if the robot gets close to the target object
- A CSV file containing the raw output values in case other visualizations are required

Figure 4.2 shows an example of the overview plot (left) and the heatmap (right).

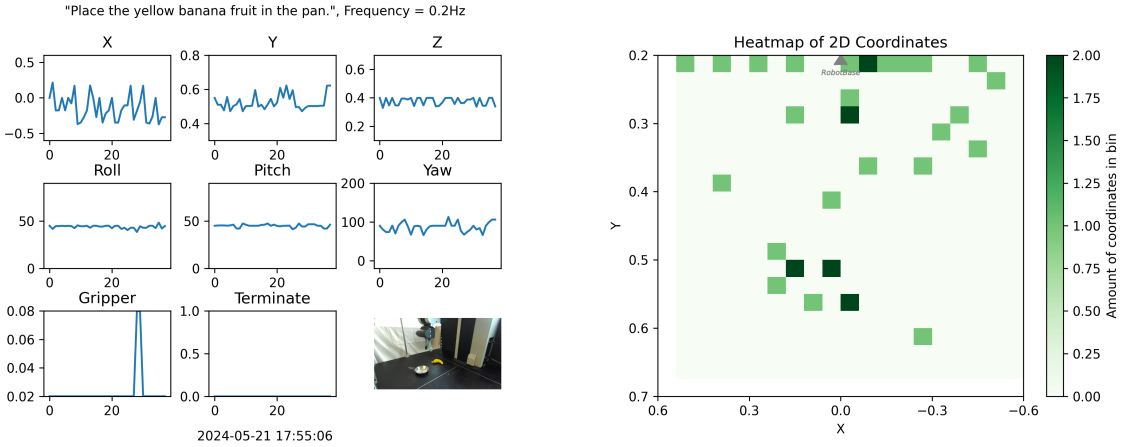

Figure 4.2: Two visualizations are generated for every evaluation run, to record behavior of the model and compare between different runs: Output plots (left) and heatmap (right).

4.3.2 Definition of Success

Before starting the experiments, it is important to define how success is measured, and what to look out for in the outcome of the experiments. The definition of a successful experiment is simple: The outcome of an experiment will be considered a success, if the robot successfully executes the task it was given. Each experiment will be ran for around four minutes, or 50 steps of inference. If the robot does not fulfill the task within that timeframe, the run is classified as "failure".

However, as early experimentation has indicated that successful task execution by the model is not likely, classifying experiment runs into "success" and "failure" alone does not offer much value. It is important to also check for behavior that "goes in the right direction", meaning it executes some of the steps necessary for successful task execution, but not all of them. This allows to get a deeper understanding of the "failed" attempts, and to separate productive behavior from complete failures.

All evaluation tasks within the experiments are of pick-and-place nature, meaning the first step towards a successful execution always consists of picking up a target object. This pick up can be further separated into steps that always need to be executed in any case: First, the robots end effector needs to move towards the target object. Then, the gripper needs to be open and positioned so that the target object is between its two sides. Finally, the gripper needs to be closed and the end effector moved upwards. Then, and only then, the pickup process is successful.

To evaluate tasks that are not fully completed, the level of completion of this series of steps can be considered. For each failed experiment, it can be determined how many of the required sub-goals were reached:

1. End effector is moved towards the general area of the target object
2. Gripper is opened
3. Gripper is moved to the exact area of target object
4. Gripper is closed
5. End effector is moved upwards

Notably, these goals must be fulfilled exactly in this order to qualify as signs of success, as e.g. moving the end effector upwards before any of the other steps are executed is not a meaningful attempt.

Experimentation showed that different variables led to different amounts of movement: In some runs, the robot made movements spanning the entire range of motion, in others, it stood completely still. While no statement about performance can be made just from the amount of movement, this will still be recorded in the testing protocol.

4.3.3 Selection of Experimental Runs

With the experimentation process established and the key variables identified, the next step is to determine the specific set of experiments to be conducted. While testing many different values for each variable would be ideal, the time-intensive nature of each experiment necessitates a more focused approach. For instance, 50 steps of inference at 0.2 Hz require over four minutes per run, making exhaustive

testing impractical.

To balance thoroughness with feasibility, two values were selected for each variable, with the exception of camera perspective. Early experiments indicated that camera perspective has the most significant impact on model behavior; therefore, three different camera angles were chosen for testing. Each combination of variable values will be tested against all other combinations, resulting in a total of 48 experiments.

The results of the experiments will be presented in Chapter 6, as well as discussed and connected to the research questions in Chapter 7

Chapter 5

Improving RT-1-X for the UMI Robot

5.1 Introduction

The experiments described in the previous chapter have investigated the ability of RT-1-X to perform on new embodiments without any additional training. Similar large, Transformer based foundation models from various domains have shown that minimal fine-tuning on domain-specific data can significantly improve the performance of pre-trained foundation models in new domains [11, 60, 61]. The ability to leverage existing data like the Open X-Embodiment dataset to reduce the required amount of embodiment-specific demonstrations would be highly relevant in robotics, where gathering data is costly and tedious.

The aim of this chapter is to investigate this, directly addressing **RQ2**: *Can fine-tuning RT-1-X on a small number of demonstrations from a new robotic embodiment improve its performance, and what are the benefits of using a pre-trained model?*

The structure of this chapter is the following: First, the process of creating a dataset of UMI demonstrations is described, starting with the selection of a task and environment setup, establishing manual robot control, and storing the data. The next section is about the training process: A fine-tuning pipeline is built around the pre-trained RT-1-X model, different hyperparameters are discussed. Two more models are trained in the course of this section, which shall serve as performance comparisons: A version of RT-1 exclusively trained on UMI data, and a version of the more recent pre-trained Octo model [62], fine-tuned on UMI data. Finally, the last section describes the experiments that will be conducted with the fine-tuned model and the baseline models, and how they address the research question. The results of these experiments can be found in Chapter 6, and will be discussed thoroughly in Chapter 7.

This chapter will deal with multiple models and versions of those models that are quite similar to each other. For reference, an overview of them is provided in Table 5.1. Their purpose will be explained in the course of this chapter.

Name	Model Architecture	Initial Training	Fine-Tuning
RT-1	RT-1	none	none
RT-1-UMI	RT-1	UMI	none
RT-1-X	RT-1	Open X-Embodiment	none
RT-1-X-UMI	RT-1	Open X-Embodiment	UMI
Octo-X-UMI	Octo	Open X-Embodiment	UMI

Table 5.1: Many different combinations of model architecture, pre-training data and fine-tuning data are used within this thesis, to compare the influence of different aspects on performance. Note that the names contain all information about the respective training data ([Architecture]-[Pre-Training]-[Fine-Tuning]).

5.2 UMI Dataset

The first step to fine-tuning RT-1-X on the UMI embodiment is to gather a set of demonstrations on the robot. This section describes the process of creating the dataset, going into detail about the task that was demonstrated and the environment it was demonstrated in, how the robot was manually controlled, and how the data was processed and saved for further training use.

5.2.1 Demonstration Scenario

To evaluate the fine-tuning potential of RT-1-X, it was fine-tuned on a single scenario from the UMI environment, meaning one task, one object, and one workspace setup. As with the zero-shot evaluation, object and task are chosen based on the ones that are already represented in Open X-Embodiment: The task is simply picking up an object from the workspace, the target object being a banana, which is amongst the most common items in Open X-Embodiment

For the workspace setup, values are chosen that led to large amounts of movement in the zero shot evaluation runs. As discussed in Chapter 4, large amounts of movement in a trial run do not necessarily indicate better model performance over a run with little movement, however it increases the chances of observing patterns in the movement. Therefore, the "Side" camera position is chosen (see Section 4.2.1). For simplicity during demonstration recording, absolute movement values are used for fine-tuning.

5.2.2 Robot Teleoperation

To demonstrate task execution on the UMI, a mode for manually controlling the robot needs to be established. While the UMI control software provides a graphical user interface that allows manual control of the robot, executing tasks repeatedly with this mode of operation is not intuitive and takes a long time. For the purpose of task demonstrations, the robot will therefore be controlled with a gamepad in the form of a PlayStation controller, which allows for intuitive, efficient control. A further discussion of the teleoperation strategy along with a comparison to other datasets from Open X-Embodiment can be found in Appendix B.1.

5.2.3 Data Processing

On the software side, the aim was to collect demonstrations that are similar in format to the datasets in Open X-Embodiment, in order not to increase the embodiment gap further than necessary. The previously mentioned input delay of the UMI also adds some challenges here.

The aim was therefore to create episodes of around 30 steps, which is the average of BridgeData, the biggest relevant dataset [59]. To achieve that with the UMI robot, which is very slow in movement compared to other embodiments, actions were logged every five seconds, or at a frequency of 0.2Hz.

It is to be noted that due to the mode of teleoperation, the gathered action data is much less "organic" than with VR controls. With the UMI and gamepad, the mode of operation is giving a control input, waiting for the robot to execute it, and then giving the next control input, while with VR teleoperation as in the bridge dataset, the control input is much more dynamic, as the robot executes it almost in real time.

After each demonstration run, an episode is saved in a simple NumPy file format. RT-1-X uses Google's RLDS dataset format. One of the researchers behind Open X-Embodiment published a tool to build such datasets[1], this code was adapted and used to convert the demonstrations to the correct format.

5.2.4 Validation

To prevent any issues that stem from false or inaccurate action recording, code was written that loads the demonstration episodes one-by-one and replays the actions in a simulated environment. This allowed to inspect each demonstration run and to make sure that the correct movements were stored.

[1] `https://github.com/kpertsch/rlds_dataset_builder`

5.3 Fine-Tuning RT-1-X

The demonstration dataset is now ready to be used for fine-tuning. This section explains how it was used to fine-tune the pre-trained RT-1-X model. It describes the development of a fine-tuning pipeline based on the available training code and discusses various parameters of the fine-tuning process. Furthermore it describes the training of two additional models that shall be used as baselines for performance comparison: On one hand, the non-pre-trained RT-1 model will be trained exclusively on the UMI dataset, which means that the Open X-Embodiment pre-training is ablated. On the other hand, the more recent Octo model architecture, also pre-trained on Open X-Embodiment, is fine-tuned on UMI data, in order to ablate the RT-1 architecture.

5.3.1 Building the Fine-Tuning Pipeline

While there is no fine-tuning code available, the authors of Open X-Embodiment provide an example of the code they used to train the model from scratch, with the entire Open X-Embodiment dataset. This code was used as a basis for finetuning.

There are several approaches to fine-tuning Transformer based models that only fine-tune certain layers, or even add additional layers to the pre-trained model that are then fine-tuned [62]. For this thesis, it was experimented with freezing the first Transformer layers and fine-tuning only the last 3 of the total 8 Transformer blocks. This did not seem to make a difference in outcome, it was decided to fine-tune the whole model for further experimentation.

5.3.2 Hyperparameters

There are multiple variables that have an influence on the training process, for which it is necessary to find suitable values. As a baseline, the values from the inital training code are used. By changing the values based on suggestions from literature, it is investigated if the fine-tuning process can be optimized.

Batch size

A batch size of 1024 was used during the Open X-Embodiment pre-training. Using a batch size of this size is not possible due to memory constraints: The machine available for training has a vRAM of 24GB. Additionally, it is common practice when fine-tuning foundation models to use a batch size much smaller than the one used during the initial training, as the fine-tuning dataset is much smaller [61]. By experimentation, it was found that the largest possible batch size on the available hardware is five episodes. This is because each episode contains multiple images, together with language embeddings and output actions.

When comparing the training process with a batch size of 2 and a batch size of 5, it shows that the training is quicker and more constant with a bigger batch size. Figure 5.1 shows a comparison of the training loss curves. It shows much less noise in the training curve of the higher batch size, suggesting that training could be further improved with a higher batch size.

Learning rate

One of the main parameters of the training process is the learning rate. RT-1 uses the Adam optimization algorithm [63] for training. One of the main contributions of this algorithm is the dynamic adaptation of the learning rate, which leads to it typically requiring little tuning of the hyperparameters [63]. In the published RT-1-X training code, the learning rate is set to 1e-4. Looking at other Transformer-based foundation models, the learning rate for fine-tuning is typically lower than the one used for initial training [64]. It was experimented with a variety of different learning rates, the training loss curves of some of them are shown in Figure 5.2. Ultimately, a learning rate of 5e-6 was chosen, which led to the most success.

Training Steps

Finally, it is important to train the model for a suitable amount of steps, allowing it to learn the behavior of the task without overfitting the training data. Through experimentation and comparison to similar fine-tuning setups, it was established that the best performance could be achieved at around 50.000 steps. Notably, the training process is quite noisy due to the encountered batch size limitations. To achieve the best possible performance, a training checkpoint was saved every 1000 training steps. Before running

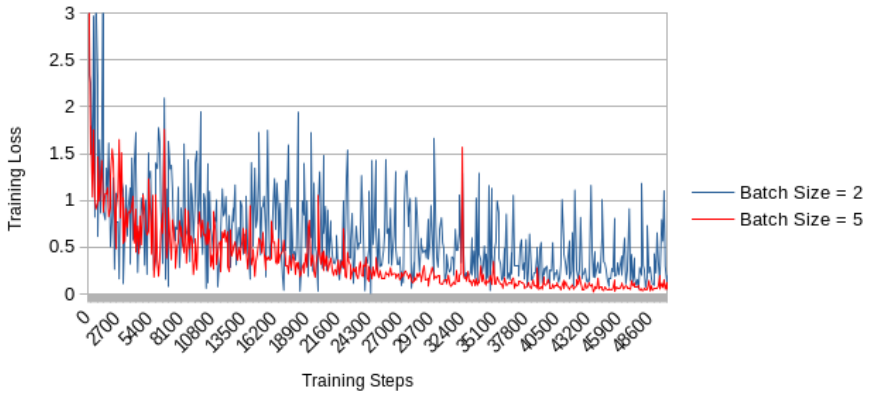

Figure 5.1: Using different batch sizes in training had significant effects on the noise in the loss curves. Compared here are training loss curves for batch size 5 (red) and batch size 2 (blue), with all other parameters unchanged. The biggest usable batch size in this research is five, due to hardware limitations.

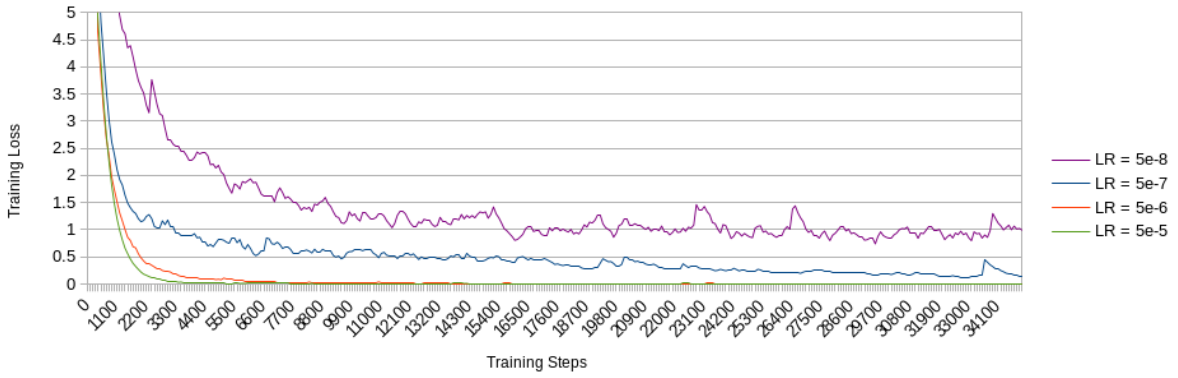

Figure 5.2: The choice of the learning rate proved to be essential for the quality of the training process. The chart shows the comparison of different learning rates when fine-tuning RT-1-X on the UMI dataset. A learning rate of 5e-6 was ultimately chosen.

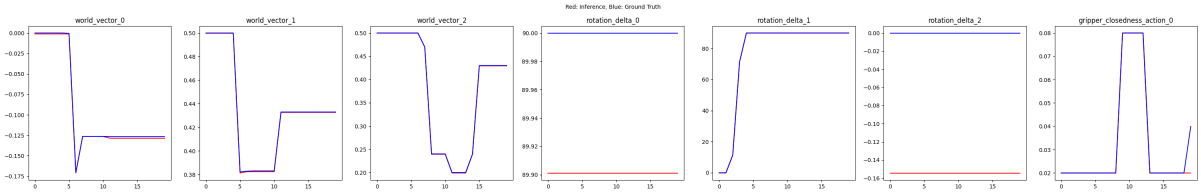

Figure 5.3: Inference is run with images from the UMI fine-tuning dataset to verify that fine-tuning was effective. Output of RT-1-X run with images from the UMI dataset is shown in red, compared to actions recorded during demonstration (ground truth) in blue.

the experiments, all checkpoints from step 40.000 to step 60.000 were evaluated. The best checkpoint was found to be at 55.000 training steps, this was then used to run the experiments.

5.3.3 Validation

To verify that the fine-tuning process was successful, inference is run with the images and language instruction from an episode of the training dataset. Figure 5.3 shows the original actions in the recorded episode (blue) and the predicted actions by the fine-tuned model (red).

5.4 Baseline Models

Two baseline models will be trained with the UMI dataset, allowing a further analysis of the performance of RT-1-X-UMI in two different aspects:

- **RT-1, trained on UMI data:** This version of RT-1 is not trained on Open X-Embodiment, but exclusively on the UMI dataset. Comparing it to RT-1-X-UMI will give insights into the benefits that can be achieved with Open X-Embodiment pre-training compared to training the same model from scratch.
- **Octo, pre-trained on Open X-Embodiment, fine-tuned on UMI data:** With the same training process and data as RT-1-X-UMI, this comparison will allow discussion and comparison of the RT-1 model architecture itself.

5.4.1 RT-1 trained on UMI Data

The from-scratch training pipeline given in the Open X-Embodiment repository is used, all hyperparameters are left unchanged with the exception of the batch size, which is set to the hardware-limited maximum of five.

Training is run for 100.000 training steps, where the loss curve reaches a clear convergence and training loss values are similar to the ones of the fine-tuning. As with the fine-tuning process, the best checkpoint between 90.000 and 110.000 steps is manually evaluated.

5.4.2 Octo fine-tuned on UMI Data

The Octo model architecture was presented in 2024 by Deepmind. It has many similarities to RT-1, although it focused much more on adaptability and tunability, and has been demonstrated to work well in fine-tuned settings. Octo will be further discussed in Section 8.1.2. A ready-to-use fine-tuning pipeline for the Octo model is provided by the authors. This pipeline is taken as-is, with the only exception of the batch size: It was found through experimentation that the maximum usable batch size constrained again by hardware limitations is 8.

5.5 Experiment Design

With the RT-1-X model now fine-tuned on a number of demonstrations from the UMI environment, this section describes the process of running the experiments necessary to evaluate the performance of the fine-tuned RT-1-X-UMI on the UMI robot. The results from the experiments can be found in Section

6.2 as well as a thorough discussion of them in Chapter 7. The evaluation can be separated into three general categories:

- **In-distribution Performance:** How does the model perform on the task it has been fine-tuned on?
- **Generalization/Transfer Performance:** Does the model generalize to new tasks and objects that were seen during pre-training, but not during fine-tuning?
- **Ablations/Baseline Comparisons:** How does the performance compare to other model setups?

The impact of fine-tuning on model performance can be evaluated by comparing the results with the ones of the zero-shot evaluation in Chapter 4. To assess also the impact of pre-training on model performance, all experiments will be also executed with a version of RT-1 that has been trained exclusively on UMI data.

5.5.1 In-distribution performance

The aim is to find out how well RT-1-X performs on the task it has been fine-tuned on, as well as to investigate if the model benefits from the initial training on Open X-Embodiment. The banana pickup task is ran 30 times, in the exact same configuration as in the training dataset. The position of the target object is changed between each experiment so that the positions are evenly distributed along the entire workspace. This is to prevent bias in the results that could stem from using "successful" positions multiple times.

Each experiment is run until the model outputs the signal to terminate the episode. An evaluation run is interrupted and counted as failure after 50 steps of inference if there is no progress towards the goal recognizable. The run is counted as a success if the robot picks up the banana from the workspace and then terminates the episode. Additionally, complete failures and "near miss" failures are recorded in the testing protocol. An evaluation run is counted as "near miss" if the robot executes a correct pick up procedure (opening gripper, moving down to workspace, closing gripper, moving up) where the X and Y positions are not accurate enough, meaning it is away a maximum of 5cm from the banana.

This experiment is also run 30 times with the RT-1-UMI model, to achieve a comparable performance baseline. The results of these experiments are shown in Chapter 6. A complete testing protocol with descriptions of notable observations can be found in Appendix B.

5.5.2 Generalization/Transfer performance

The goal is to evaluate if RT-1-X-UMI is able to generalize to new objectives on the UMI by leveraging knowledge learned on other embodiments, during Open X-Embodiment pre-training. This will be evaluated in three dimensions. Each dimension will be tested with one experiment setup:

- **Generalization to new objects:** The same pick up task will be run, only this time with an apple instead of the banana. The apple is the most common food object in Open X-Embodiment.
- **Generalization to new tasks:** The new task will be "Place the yellow banana in the pan.". Pick-and-place tasks are very common in Open X-Embodiment, as well as the pan object.
- **Generalization to environments:** The color of the workspace table will be white instead of black.

Each of these experiments will be conducted ten times, with the target object in random positions on the workspace. For the generalization to new objects and new environments, "near miss" attempts will again be recorded. For the new task, near miss attempts are harder to classify, as the task is more complex and there are multiple stages where accuracy is required (pick up, placement). This experiment will therefore be rated as either fail or success, although detailed observations about the robots exact behavior are logged in the test protocol in Appendix B.2

Chapter 6

Results

The following chapter presents the experimental results obtained during the course of this research. These results are structured to address the key research questions outlined earlier, focusing on the performance and adaptability of the RT-1-X model when applied to new robotic embodiments, particularly the UMI robot.

The chapter begins by presenting the outcomes of the zero-shot experiments, which evaluate the model's ability to generalize to a new embodiment without any additional training. This is followed by a detailed examination of the results from the fine-tuning process on the UMI robot. This is done by first addressing the model's performance on tasks specifically demonstrated in the fine-tuning dataset, and then exploring the potential for transferring knowledge learned during the Open X-Embodiment pre-training to the new embodiment. Additionally, the sensitivity of the model to changes in the environment, as compared to the original fine-tuning conditions, is analyzed. Finally, the chapter investigates the overall impact of the Open X-Embodiment pre-training and the influence of the RT-1 model architecture on the model's performance. Full procotols of the experiments can be found in Appendix B.2.

Table 6.1 presents an overview of all the experiments conducted, listed in the order in which their results are presented in this chapter. Each section provides both quantitative and qualitative analyses of the experimental data, highlighting key patterns, successes, and limitations observed during the testing process. The results presented here form the foundation for the subsequent discussion, where these findings are interpreted and contextualized within the broader scope of robotic learning and model adaptation.

6.1 Zero-Shot Experiment Results

In the first step of this research, the performance of the RT-1-X model zero-shot on the unseen UMI embodiment was investigated. As first trial runs indicated that no successful task executions were possible, the focus of experimentation was laid upon investigating the effects different environment

Experiment	Relevant RQ	Model Version	Evaluation Objective
EX1	RQ1	RT-1-X	Zero-shot generalization to new embodiment
EX2	RQ2	RT-1-X-UMI	Performance on fine-tuning scenario
EX3 (a&b)	RQ2	RT-1-X-UMI	Object knowledge transfer from Open X-Embodiment to fine-tuning domain
EX4	RQ2	RT-1-X-UMI	Task knowledge transfer from Open X-Embodiment to fine-tuning domain
EX5	RQ2	RT-1-X-UMI	Sensitivity to changes in environment
EX6	RQ2	RT-1-UMI	Benchmark performance without Open X-Embodiment pre-training
EX7	RQ2	Octo-X-UMI	fine-tuning performance of RT-1 architecture

Table 6.1: Many different experiments were conducted to evaluate different aspects of generalization performance.

factors had on performance and general model behavior.

The RT-1-X model was evaluated on the UMI embodiment in a variety of environment setups, involving different objects, backgrounds, camera perspectives, tasks, and instructions. A series of 48 structured experiments was conducted. The result of these experiments were zero successful pickup attempts. The model behavior can be described as random in all cases, showing no clear attempts by the robot to reach its intended goal.

No performance improvements could be observed by changing environment variables. Some changes in general behavior were observed when changing certain variables: While for some setups the model output very similar values at each inference step, other setups showed much more variation in output between each step. This was especially prominent when switching between the camera positions "Side" and "Shoulder": Evaluation runs with the "Side" camera position showed significantly more variation with the model outputs spanning almost the entire action space on some axes, while for the "Shoulder" camera position, outputs are always very close to zero (or the middle of the robot action range after conversion). Figure 6.1 shows output logs of trial runs with the two camera positions.

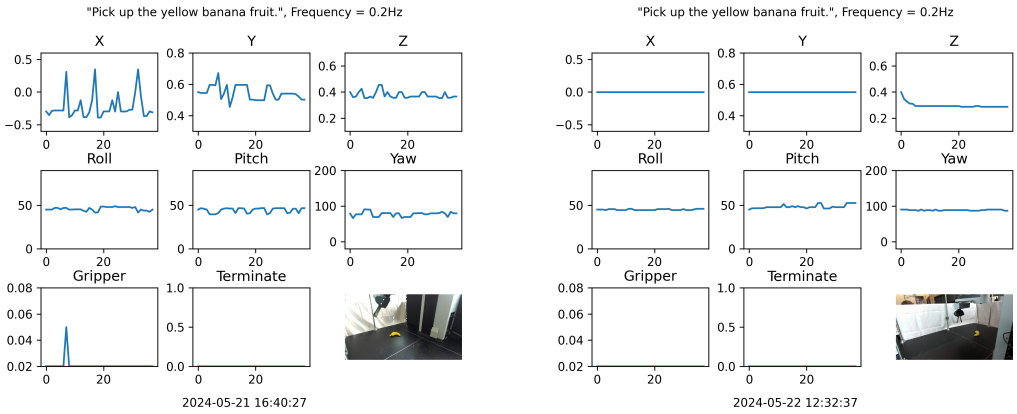

Figure 6.1: Camera position was found to have the biggest effect on model behavior. Camera position "Side" (left) shows significantly more variation than "Shoulder" (right).

Despite this observation, the results of the experiments with the zero-shot model are clear: The RT-1-X model shows no successful knowledge transfer to the unseen UMI embodiment.

6.2 Fine-Tuning Results

The second step of this research was to investigate if the performance could be improved by fine-tuning the model on demonstrations from the UMI robot. The RT-1-X model was fine-tuned with a dataset of 100 manual demonstrations of the UMI executing a simple task (picking up a banana from the workspace).

6.2.1 In-Distribution Performance

The first experiment evaluated the performance of the fine-tuned model on the demonstrated task. In this experiment, a success rate of 23% could be achieved. In most of the non-successful evaluation runs, the robot executed the correct movements, and was off from the target object by less than five centimeters. Experiment runs where the robot executes the correct sequence of actions and is off by less than five centimeters are classified as "near miss". All evaluation runs that resulted either in success or a near miss combined make up 80% of all evaluation runs. Table 6.2 shows the outcome of the experiments.

Observation	Amount
Success	7
Near Miss	17
Failure	6
Total	30

Table 6.2: Results of the fine-tuned model on the demonstration task

Notably, the model shows signs of being biased towards certain positions on the workspace. When repeating a near miss run with the target object position unchanged, the robot typically executes a near miss in the exact same position. When moving the target object to this position, the robot is able to successfully pick it up. Possible reasons for this behavior are discussed in Chapter 7.

6.2.2 Transfer and Generalization

RT-1-X has been shown to transfer knowledge across embodiments within its pre-training dataset, as well as to generalize well to changes in environment. Experiments were conducted to evaluate if these abilities can be extended to the new embodiment through fine-tuning.

Transfer of Object Knowledge

The first experiment aimed to investigate if objects that were only seen in Open X-Embodiment during pre-training are recognized on the UMI, by the fine-tuned model. A coke can, which is a common object in Open X-Embodiment, was placed on the workspace and the model was instructed to pick it up. As a second part to this experiment, the coke can and the original banana were placed on the workspace, the robot had to decide which one to pick up.

Table 6.3 shows the results of the experiment with only the can present in the workspace. A success rate of 10% and a success or near miss rate of 60% was achieved. When both the coke can and the banana object were placed on the workspace, the results were significantly worse, with no successful pick ups. To capture the object recognition abilities of the model, these experiments are classified as either near miss attempts on the coke can, near miss attempts on the banana, or no significant attempts at either of the objects. The results of this are shown in Table 6.4.

Observation	Amount
Success	1
Near miss	5
No attempt	4
Total	10

Table 6.3: Results of can pick up with only can in workspace

Observation	Amount
Coke pick up attempt	5
Banana pick up attempt	3
No attempt	2
Total	10

Table 6.4: Results of can pick up with can and banana in workspace

Transfer of Task Knowledge

To investigate the transfer of task knowledge to the new embodiment, the model was prompted to place the known target object into a cooking pan. No meaningful attempts of executing the task successfully were observed. In most of the cases, the robot executed pick up attempts for either the banana or the pan, and terminated after the pick up attempt, no effort was made to place the object somewhere else.

Observation	Amount
Pan pick up attempt	7
Banana pick up attempt	2
No attempt	1
Total	10

Table 6.5: Results of banana-in-pan task

Sensitivity to Environment Changes

To investigate how the model handles changes to the environment, the environment observed by the model was altered: The black workspace was covered with a white tablecloth. This lead to a dramatic loss of performance, with the model only achieving one near miss attempt, all other trial runs resulted in either random movement or pick up attempts far away from the target, as shown in Table 6.7.

Observation	Amount
Success	0
Near miss	1
No attempt	9
Total	10

Table 6.6: Results of changing the workspace background

6.2.3 Ablations

The RT-1-X foundation model consists of two main components: The RT-1 model architecture, and the Open X-Embodiment pre-training. The following two experiments were designed to ablate each of those components, to investigate the roll they play in the performance and training efficiency of RT-1-X-UMI.

Ablating Open X-Embodiment Pre-Training

To evaluate the effect that the Open X-Embodiment pre-training has, a version of RT-1 is trained exclusively on the UMI dataset, which allows for comparison of the learning process as well as of the performance.

The training process shows that the behavior on the UMI embodiment can be learned much faster with Open X-Embodiment pre-training. To compare the performance, the same evaluation on the demonstrated task as before is run with the from-scratch model. This version of the model shows significantly less success than the fine-tuned variant, with a success rate of 10% and an accumulated success or near miss rate of 33%.

Observation	Amount
Success	3
Near miss	7
No attempt	20
Total	30

Table 6.7: Results of RT-1 trained on UMI without pre-training

Ablating the RT-1 Model Architecture

To investigate the fine-tuning suitability of the RT-1 architecture, it is compared to another model architecture which has been proven to be well suited for fine-tuning on new embodiments, the recently introduced Octo model. It is similarly pre-trained on Open X-Embodiment, and fine-tuned on the UMI dataset. In the fine-tuning process, this model proved to be more resource efficient, allowing to use a larger batch size, which also led to a less noisy training process.

When evaluating its performance on the demonstrated UMI task, this model showed good performance on object recognition and location, however no pick up attempts were observed, as the robot never opened its gripper or moved low enough to the workspace. This however is likely a problem in the training setup, this issue will be further discussed in Chapter 7. The results for this experiment are therefore not counted as valid, and not used for comparison.

	EX2: fine-tuning Task	EX3a: Object Transfer: Single Object	EX3b: Object Transfer: Two Objects	EX4: Task Transfer	EX5: Environment Change	EX6: Without Open X-Embodiment	EX7: Octo Model*
% Success	23	10	0	0	0	10	-
% Success + Near Miss	80	60	50	0	10	33	-
% Total Failures	20	40	50	100	90	66	-
# of Trial Runs	30	10	10	10	10	30	-

*Likely issues in training of Octo, results not comparable

Table 6.8: A comparison of the results of all experiments conducted. For additional results that are not comparable across all experiments, see the respective sections in this Chapter.

Chapter 7

Discussion

Following the presentation of methodology, experiments and results in the previous chapters, this discussion synthesizes the findings, relates them back to the research questions, and reflects on their broader implications within the field of robotic learning. The main part of this Chapter explores the results of the experiments, and based on them, critically examines the performance and limitations of the RT-1-X model. This will be organized around the key research questions posed in the thesis. Secondly, additional findings that were made in the course of research are presented. While they do not relate directly to the research questions, they still hold valuable information about the process and its components. Lastly, threats to the validity of the findings and limitations of this research are discussed.

7.1 Main Findings

Can the RT-1-X model generalize to a completely unseen robotic embodiment, without any additional data, utilizing knowledge learned from the Open X-Embodiment dataset?

RT-1-X has previously been shown to generalize very well to unseen objects, backgrounds and environments, as well as the ability to transfer tasks learned on one embodiment to other embodiments within the training data [1]. Generalizability to unseen embodiments however has not been shown yet. The aim of the first part of this thesis was to find out if RT-1-X is able to generalize to new embodiments out-of-the-box. A complete control pipeline was build around RT-1-X for the UMI robot, including robot control logic, infrastructure to communicate between the model and the robot, and the inference code itself. All parts of the pipeline were carefully tested and validated, to eliminate any issues that could distort the outputs of RT-1-X.

The zero-shot performance of RT-1-X was found to be lacking. Early experimentation revealed that the model's performance was far from achieving successful task execution. As a result, the focus shifted toward systematically experimenting with all factors of the robotic environment to understand their influence on model behavior and to explore whether any specific setup could facilitate progress in task execution. Over 200 exploratory trial runs were conducted, followed by a set of 48 systematic experiments. Despite these extensive efforts, no successful task executions were observed. Although the experiments were designed so even small progress towards the goal would be reflected in the results, no behavior that suggested any meaningful attempt to reach the goal was observed.

Altering certain environmental factors had one noticeable effect on the model's behavior: in some configurations, the model produced significantly greater variety in its outputs, resulting in more robot movement. This raises the question of whether increased movement necessarily indicates a better-performing environment setup compared to scenarios with minimal movement. No discussion about this could be found in literature. Despite further experimentation with the configurations that generated the most movement and attempts to connect this increased variation in model output to existing research, no definitive conclusions could be drawn. Consequently, it can be stated that despite this observation, no improvements in task performance were observed through extensive experimentation with environmental setups.

This leads to the following clear finding addressing **RQ1**:

Finding 1: RT-1-X is not able to generalize any of the knowledge learned on Open X-Embodiment to the unseen UMI robotic embodiment out-of-the-box.

Since zero-shot generalization was tested on only one embodiment (the UMI), it remains possible that RT-1-X could perform effectively on other embodiments out of the box. However, this example proves that RT-1-X is not universally generalizable to new embodiments.

The lack of real change in model performance when altering multiple environmental factors during experimentation makes it difficult to narrow down the specific aspects of the new embodiment that cause the failure. As RT-1-X has previously shown strong generalization to new objects, tasks, environments, and distractors, this finding calls into questions which factors of the embodiment gap are responsible for the complete lack of zero-shot performance. The fact that RT-1-X handles significant environmental changes yet fails entirely in adapting to a new embodiment also leads to further discussion on the fundamental differences between adapting to a new environment versus a new embodiment.

Although, strictly speaking, an embodiment refers to the robot's physical form, including its sensors and actuators, in practice, the embodiment and its environment are closely intertwined. The "environment" observed by the robot's camera includes not only the workspace, background, target objects, and possible distractors but also the robot's own embodiment as a visible part of the scene. In these experiments, efforts were made to replicate many factors from the datasets used in Open X-Embodiment and avoid introducing additional distractors. The camera perspectives and objects were selected based on common examples from Open X-Embodiment, and neutral backgrounds were chosen to mirror typical setups in the pre-training dataset. Nevertheless, significant differences remain in the environment that are not strictly related to the embodiment itself.

When looking at the robotic embodiment itself, the main difference between the UMI and all other embodiments from Open X-Embodiment is the robot type: The UMI is a robotic arm of the SCARA type, no other robots of this type are present in Open X-Embodiment. RT-1-X is built to control robots directly via end effector coordinates, meaning the difference in control logic between SCARA and other robots are not within the scope of the model. However, a SCARA robot is different in appearance from other embodiment types, and movements are executed differently. The same end effector position can therefore look very different on the image that is used as model input.

While these are some dfferences that could be part of the reason RT-1-X is not able to generalize to the new embodiment, narrowing down the concrete factors that differentiate embodiment transfer from simple changes in environment is an objective for further research.

RQ2: RQ2: Can fine-tuning RT-1-X on a small number of demonstrations from a new robotic embodiment improve its performance, and what are the benefits of using a pre-trained model?

Given the failure of zero-shot performance, the research shifted focus to whether RT-1-X could be fine-tuned to work effectively on the UMI robot. While achieving zero-shot generalization to new embodiments would be highly significant as it would enable universal robot control by RT-1-X, the ability to fine-tune with minimal effort is still highly valuable, as it allows a robot to learn a large number of skills without an extensive data gathering and training process. The experiments explored gathering demonstrations on the UMI robot and the fine-tuning process, the impact of pre-training on Open X-Embodiment, and the transferability of task and object knowledge from the pre-training phase to the fine-tuning domain.

RQ2.1: How does the fine-tuned RT-1-X model perform on the specific task for which it was fine-tuned?

When evaluated on the in-distribution task, the fine-tuned model shows a success rate of 23% over 30 experiment runs. This is a significant achievement compared to the non-existent zero-shot performance and shows that it is possible to further improve the RT-1-X model for specific embodiments by fine-tuning.

This number is however still low when compared to the performance of RT-1-X on embodiments that were part of its pre-training. A comparison between the success rates of pre-training environments and the fine-tuned UMI results is given in Table 7.1. Also listed is the number of demonstrations that each dataset consists of. It is probable that with a higher number of episodes, the success rate of the fine-tuned scenario would also increase.

Setting	Episodes	Success Rate
RT-1-X-UMI on UMI robot	100	23%
Jaco Play	976	63%
Berekley Cable Routing	1482	56%
NYU VINN	435	80%
Berekley Autolab UR5	896	45%
Freiburg Franka Play	3242	72%
Berekley Bridge	25460	27%
Original RT-1	73499	73%

Table 7.1: RT-1-X shows varying amounts of success even within setups from its training data. RT-1-X-UMI scores the lowest, however it also has the lowest amount of episodes in its dataset. [1]

When looking at the failed evaluation runs, the test results show that 77% of all failed attempts are near misses, meaning the robot executed the correct sequence of actions to pick up the target object, however it was off on the X and Y coordinates by a small margin. This means that 80% of evaluation runs led to a result of either success or near miss, showing that while the model managed to pick up the behavior patterns of the UMI robot, it is struggling with the accuracy of its movements.

A possible reason for this is the method of collecting the training dataset. The manual demonstrations were conducted by looking directly at the robot and not at the camera image, meaning the position of the gripper could be observed from multiple perspectives, allowing for far more accurate positioning than with only the camera perspective. Furthermore, no "corrections" were part of the training data: When the robot, controlled by the model, starts a pickup attempt which is off by only a few centimeters, it does not know how to correct the position, as all pickup attempts in the training data were accurate on the first try. This lack of knowledge about correcting mistakes and inaccuracies is well known in the field of imitation learning.

An interesting phenomenon observed during experimentation is that the robot apparently has "preferred" pickup spots. These are specific target object locations that consistenly lead to successful pick up. This behavior was confirmed through an additional experiment: After an initial experiment where the pick up attempt is a few centimeters away from the target, the target object is placed at the exact location where the robot tried to pick it up before. Subsequent trials with this new location are then typically successful. This could indicate that the model is overfitting its training data, by achieving a high success rate for locations that are part of the training demonstrations, and lower success for locations that are unseen. While overfitting could be caused by training the model too fast or too long, it was experimented with both of these aspects (see Chapter 5), no improvements were observed with slower or shorter training. This issue is therefore likely a symptom of the rather small training dataset.

Finding 2.1: The performance of RT-1-X can be improved on a new embodiment by fine-tuning, although it does not reach the same level as for the pre-training embodiments. The main issue is accuracy, with most failed attempts being less than 5cm off.

RQ2.2: Can the fine-tuned RT-1-X model transfer concrete knowledge learned during Open X-Embodiment pre-training to the new embodiment and maintain performance when faced with changes in the demonstration environment?

To take the investigation of the effects of pre-training one step further, it was investigated if concrete skills learned on the pre-training embodiments can be transfered to the new domain. It has been shown that RT-1-X is able to take skills that were learned on one embodiment and transfer them to another embodiment. This has however only been within the embodiments that are part of the Open X-Embodiment pre-training mix. This series of experiments investigated if tasks learned during pre-training could be transferred to the new fine-tuning embodiment.

The first experiment investigated if the model recognizes objects that were only seen in pre-training. After initial success with using a coke can as the target object instead of the banana, an additional experiment was conducted to investigate the models object recognition capabilities: When running the

pick up task with both the banana and the coke can on the workspace, the model did not show capabilities of reliably identifying the correct object. In around half the attempts, the robot tried to pick up one of the two object, the other half of the experiments did not show any significant pick up attempts. This implies two things: First, knowledge about other objects is not transferred from the pre-training data. The second, unexpected observation is that the model is also not able to reliably identify the object it has been fine-tuned on as soon as there is another object on the table. The model has only learned to pick up an object from the workspace, but has no further understanding of what the object is.

When investigating the transfer of new tasks to the UMI, a similar observation can be made. The task is to place the banana in the pan, while on the workspace there is the banana and a cooking pan. As in the previous experiment, the model tries to pick up one of the objects in around half of the attempts. This time it is almost always the pan that the model tries to pick up. This shows again that it has no capabilities of object recognition. Furthermore, after the pick up attempt of either the banana or the pan, the model sends the termination signal, it makes no effort to place the object somewhere else. This shows that it has no understanding of the new task, and only executes the pick up task it has been fine-tuned on.

To investigate the effect of changes in the environment, the black workspace is covered with a white tablecloth while leaving everything else about the setup as-is. This led to a dramatic decrease in performance, with only one significant pick up attempt, and all other trial runs resulting in random movements. This indicates that the fine-tuning process adapts the model very closely to the fine-tuning domain. Minor changes to the environment making the scene appear differently than in the demonstrations instantly disrupt the performance.

Finding 2.2: Concrete knowledge about tasks and objects learned during pre-training is not transferred to the new embodiment during fine-tuning, and slight changes to the environment reduce performance drastically. Only the skills learned during fine-tuning, in the environment used during fine-tuning, can reliably be reproduced.

RQ2.3: What benefits does pre-training on the Open X-Embodiment dataset provide when fine-tuning RT-1-X on a new embodiment?

The previous discussion has shown that the model is generally able to be adapted to the UMI robot. It has however not been investigated yet what the actual benefit of fine-tuning a pre-trained foundation model is, compared to simply training a model from scratch.

To do so, a version of RT-1 was trained from scratch and exclusively on the UMI dataset. This allows a comparison between two versions of the exact same model (RT-1), one version being a "simple" end-to-end control model trained on a specific embodiment, and the other version being the RT-1-X foundation model with Open X-Embodiment pre-training.

The from-scratch training was initially conducted using the same exact training parameters as for the fine-tuning process. Figure 7.1 shows the loss curves of the from-scratch training and the Open X-Embodiment based fine-tuning. For the fine-tuning, the training loss drops significantly faster than for the from-scratch training. This indicates that the knowledge learned from Open X-Embodiment does indeed help to learn the correct model behavior for the UMI embodiment, and can significantly speed up the training process.

To get a performance comparison, the learning rate for the from-scratch training was increased and training was run for twice as long, to account for the observed slower learning. The same exact experiments as with the fine-tuned model were rerun, leading to lower performance with a success rate of 10%, and a cumulated near miss and success rate of 33%. This indicates that not only the training speed can be improved by choosing the foundation model over training from scratch, but also the resulting performance is better.

These observations show that the use of the pre-trained foundation model is able to both speed up the training process, and improve the results that can be achieved with a small training dataset.

So far, it was found that RT-1-X can be improved for the UMI robot, although with significant limitations in accuracy and generalizability. I was also found that the Open X-Embodiment pre-training significantly improves the learning of new embodiment behavior. What hasnot been investigated so far is the influence of the RT-1 model architecture, which is the basis of the RT-1-X foundation model. An effort was made to investigate this by comparing it to the very recent Octo model, which is also available pre-trained on Open X-Embodiment. A detailed presentation of Octo can be found in Chapter 8. In comparison to RT-1-X, there is a ready-to-use fine-tuning pipeline available, which was used to fine-tune

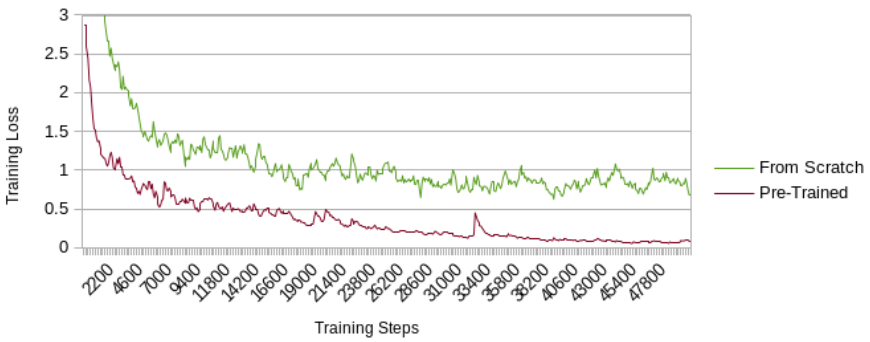

Figure 7.1: Learning from scratch exclusively on UMI data (green) is significantly slower than when using Open X-Embodiment pre-training (red). Exponential smoothing was applied to the loss curves for clarity.

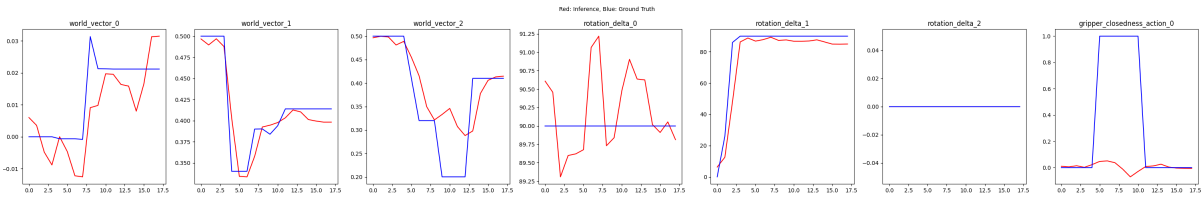

Figure 7.2: While the Octo model fine-tuned on the UMI (red) generally fits the training data (blue) well, the gripper dimension seems unaffected by training, and the Z coordinate never goes all the way down to the workspace.

the model on the UMI demonstrations.

The fine-tuning runs much more memory efficient and allows the use of a higher batch size and therefore a smoother learning process. When running inference with the fine-tuned model on sample data, the model shows very promising behavior. Overfitting to the training data seems to be less of a problem than with RT-1-X. However, the model seems to have problems with two of the output dimensions: The gripper is never actuated, and the z-axis never goes down far enough to pick up the target object. Figure 7.2 illustrates this, running inference with images from the UMI dataset: While most dimensions show accurate predictions, the z axis never goes down, and the gripper is not actuated. This behavior is confirmed when running the fine-tuned Octo model on the actual UMI robot: In all attempts, the arm moves to the correct x and y axis positions quickly and very accurately, but then does not open its gripper and does not move down to the workspace. Interstingly, while all axes seem to fit the training data more accurately the longer the training is run, the gripper coordinate does not improve, and the z coordinate actually gets worse. It was experimented with different training parameters and scaling, but this situation could not be improved.

While due to this problem the Octo model technically showed a 0% success rate, it is highly likely that this issue is training related and does not stem from the embodiment or the model architecture. Therefore, no conclusions about the performance of the Octo model can be made at this point, and the comparison of Octo and RT-1 fine-tuning is left for further reserach.

Finding 2.3: While no concrete knowledge is transferred, using the pre-trained model over training a model from scratch still offers advantages in training speed and resulting performance.

7.2 Additional Observations

In the process of working on this thesis, one of the biggest challenges was to bring a great variety of technologies together, from the 40 year old UMI robot to the state-of-the-art RT-1-X model that came out last year. This involved getting familiar with each technology and making them work individually, before finally connecting them and integrating the entire system. This revealed challenges and intricacies of the different parts of the system, the lessons learned about them shall be presented in this section.

7.2.1 Working with the UMI RTX

The UMI RTX is very different to robot embodiments that are typically used in research. Most of the differences can be attributed to its age: It was produced in the 1980s and 1990s, a very low number of UMI robots is still in use today. There are multiple challenges that were found to arise from working with such an embodiment.

The first issue is that compared to other, standardized embodiments, it was never built to be used with modern robotic tooling like ROS. Using it in modern research scenarios and together with modern components, e.g. the RT models, requires custom code to make it work. Two such custom tools were build for the UMI robot at UvA: On one hand, low-level control code, providing an interface to control individual joints of the robot. The other part is a software package building a ROS2 interface around that interface, together with code for inverse kinematics, GUI, path planning and object detection. While this package includes many features, it was build for a single research project by two students which are no longer present at UvA. The limited use of the UMI robot outside this research project (neither at UvA nor at other institutions) led to lack of testing and generalization of this software. At the start of working on this thesis, it turned out that the ROS2 interface for the UMI was much less complete than expected. Getting it to work reliably required much experimentation and debugging. The result is a new, minimal ROS2 interface for the UMI, which does not support previous features like path planning and object detection, but offers simple ROS2 controls for the UMI. There are still limitations to this, as the inverse kinematics logic does not reliably work at the edge of the robot's action space. While for this project, the robot movement was simply limited to the area where the inverse kinematics work, for future work with the UMI robot, it would make sense to rework the inverse kinematics logic.

On the mechanical side, the age of the UMI leads to issues as well. Most joints of the UMI are controlled with toothed belts, which get worn down over time. This wear is increased by two factors: The age of the belts, especially the spare parts, which have been in storage without being moved for over 40 years. This leads to the plastic material losing flexibility and becoming brittle. In the calibration procedure, the UMI moves all joints to their physical stops, causing a large force especially on the shoulder belt. Additionally, due to the design of the UMI, certain movements lead to the shoulder belt to jump, this is a known limitation of the design [58]. This leads to heavy wear on the old, brittle belts. In the course of this thesis, the shoulder belt had to be replaced three times. This leads to the next issue: The lack of available spare parts posed the additional challenge of identifying an off-the-shelf belt that would fit the UMI pinions. Through measuring the remaining spare parts and experimentation with various standardized belt sizes, a suitable replacement for the original spare part was found, guaranteeing the continued availability of spare parts for the UMI, and extending its life even further.

7.2.2 Working with RT-1-X

The public availability of the RT-1 code, the Open X-Embodiment dataset, pre-trained checkpoints and training code is an invaluable contribution to the robotics research community. However, there are certain issues that arise when trying to run the code, as well as difficulties when adapting it.

First of all, there is no formal documentation given along with the code, and comments in the code are very rare. This leads to uncertainties in the process of working with the code. There are two implementations for running inference with a trained model checkpoint, one based on Tensorflow and one based on Jax. Even before actually working with the code, it is unclear why there are two implementations, if there are differences between them, and which should be chosen. In the course of this thesis, both implementations were used to run inference, and no differences in outcome could be observed. Notably, all code that goes beyond running simple inference, e.g. the training example, is only available as a Jax implementation, which is why in the further course of this thesis, the Jax implementation was used.

Once all dependencies were set up and all parts of the code ran successfully, switching from the dummy data used in the published code to real inputs, meaning actual images and language instructions, posed the next challenges. It was not documented which image formatting the model expected exactly, requiring delving into the actual RT-1 and RT-1-X code to find out what processing steps should be applied to the images before running inference. Also, it was unclear how to handle the required history of 15 images during the first 14 steps of inference, where only a history of 1-14 images is available. Debugging both of these issues was tedious, as the only symptom of mistakes were "strange" outputs of the model, which were impossible to trace back to specific issues. Setting up the whole pipeline and verifying that the model behaved as it should was therefore a long and tedious process. However, in the end, all issues were found and mitigated, and the model inference behavior was verified by running inference with images

from the training datasets as inputs, and comparing the outputs to the actual movements.

The lack of documentation also posed challenges when creating the fine-tuning pipeline. An example of training the model from scratch on the original dataset was available, which served as the core for the fine-tuning logic. An issue that was found during the first training runs was that the training code only considered the first ten demonstrations from every dataset. It is probable that this was implemented intentionally by the authors to increase training speed and use less resources for demonstration purposes. However, there was no comment about it at the relevant part of the code, and no other mention of this anywhere else.

Another issue that was encountered when assembling and evaluating the fine-tuning pipeline was a bug in the loss calculation of the training code. This bug lead to wrong scaling of model outputs in the loss function, which in turn lead to wrongly scaled and clipped values when running inference. As this happened at a low level of the training logic, and was combined with other (intentional) scaling effects, realizing and tracking down this issue was a challenge, but the problem was found and fixed. As this is a clear bug in the provided training code, a pull request was filed on the Open X-Embodiment GitHub repository[1]. At the date of finishing this thesis, there has been no reaction from the authors yet, although the hope is that this contribution will be accepted to improve the training example.

The lack of documentation and low code readability are common problems in the open source community [65] as well as with scientific software in general [66, 67], where reproducibility of results is often not possible due to issues with the published code [68].

7.2.3 Working with Open X-Embodiment

The Open X-Embodiment datasets could also benefit from more documentation. While there is a Google sheet available that lists all datasets of Open X-Embodiment along with important variables as episode amount, embodiment type, data collect method and much more, some useful dimensions are missing. When designing the demonstration collection process for the UMI robot, it would have been helpful to have more information about the actual data recording process, meaning information about which datasets use absolute positions, relative positions or joint velocities, as well as the action space, data recording frequency (time passed between datapoints in an episode) and average episode length. Having this data would have been useful to keep the non-physical embodiment gap as small as possible. Acquiring this information from each dataset can be done, but is very time consuming [69].

7.3 Threats to Validity

Throughout this research, significant efforts were made to obtain representative and reproducible results, explore multiple perspectives on performance, and validate experiment setups and training processes at each step. However, several threats to validity remain that must be considered when interpreting the findings.

Single Embodiment Limitation. The primary objective of this research was to evaluate the model's ability to generalize to new robotic embodiments. However, this capability was assessed using only one embodiment, the UMI robot. Evaluating generalization on a single embodiment imposes significant limitations, as it does not allow for broad conclusions about the model's performance across different types of robots with varying physical characteristics and sensor configurations. While the results from the UMI robot provide valuable insights, they cannot fully represent how the model might perform on other embodiments. Nevertheless, the poor performance observed on the UMI robot, especially in the zero-shot setting, is critical, as it conclusively demonstrates that universal generalization—where the model would work seamlessly across all new embodiments without any modifications—is not possible.

Environmental and Embodiment Interdependencies. Another threat arises from the difficulty in strictly isolating differences in embodiment from environmental factors, as certain parts of the environment are inherently tied to the specific embodiment being tested. Although the scope of the investigated embodiment gap was thoroughly discussed, the inability to fully separate these factors needs to be considered when interpreting the results.

[1] `https://github.com/google-deepmind/open_x_embodiment/pull/84`

Experiment Sample Size. The number of experiments conducted could also be a threat to validity. While the main experiments were run 30 times and smaller experiments 10 times, a larger number of repetitions would provide more representative results. However, the time-consuming nature of each experiment, which requires active monitoring, limited the feasibility of running more trials. Additionally, this constraint made it challenging to test a broader range of tasks, objects, and environments, potentially limiting the comprehensiveness of the findings.

Observer Bias in Result Interpretation The interpretation of results often relied on subjective observation, particularly in assessing whether the robot made general progress toward a goal. While efforts were made to quantify these observations, the possibility of observer bias remains, which could affect the consistency and objectivity of the conclusions drawn.

Fine-Tuning Process and Demonstration Dataset. The fine-tuning process itself presents a threat to validity, as no established guidelines or examples for fine-tuning RT-1-X were available. Despite thorough experimentation with the fine-tuning pipeline and hyperparameters, fine-tuning is an intricate process with many variables. While optimizing the fine-tuning process was beyond the scope of this thesis, it is possible that the results could be further improved with a more refined approach.

Similarly, the process of recording demonstration tasks for fine-tuning poses a validity concern. Demonstrations were only collected once, for a single task, in one environment. The strategy for gathering these demonstrations also differed from other approaches due to the UMI's input delay. Given the time-intensive nature of collecting 100 demonstrations, it was not feasible to experiment with different demonstration setups, potentially limiting the diversity and generalizability of the training data.

Chapter 8

Related work

Throughout this research project, a substantial amount of literature was reviewed: initially, to gain an overview of the field; during the project, to clarify concepts and guide the process and experimentation; and after obtaining results, to put them into context and deepen the understanding. This section brings together the most important works in this area, highlighting key studies and recent developments that are relevant to this research and showing the direction in which the field is moving.

Robotic learning is a rapidly evolving area, with a constant flow of new research. Given the vast amount of work available, this literature review focuses mainly on generalization to new embodiments, as it is the main focus of this thesis. For those interested in a broader view of the field, several recent literature surveys offer comprehensive overviews of foundation models in robotics, covering a wide range of topics beyond the scope of the literature review presented in this thesis [10–13, 70].

8.1 Approaches Similar to RT-1-X

This thesis investigated the generalization of RT-1-X, a Transformer-based foundation model using imitation learning, to new embodiments. A number of similar approaches was identified, and it was analyzed which aspects of generalization they explore. To assure comparability, these are only approaches where a model is trained **end-to-end**, and with **imitation learning** using robot demonstrations. Approaches that differ from this are discussed in Sections 8.2 and 8.3. Table 8.1 shows an overview of all approaches that were found to be similar to RT-1-X. Two approaches which are especially relevant for this thesis are further presented.

Publication	**Date**	**Using Transformer**	**Gen. to new objects/ tasks/ environments**	**Zero Shot to new embodiments**	**Fine-Tuning to new embodiments**
BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning [28]	2022	No	zero shot	not tested	not tested
Polybot: Training One Policy Across Robots While Embracing Variability [71]	2023	No	zero shot/fine-tuned	not tested	not tested
Robot Learning with Sensorimotor Pre-training [72]	2023	Yes	fine-tuned	not tested	yes
Mobile ALOHA: Learning Bimanual Mobile Manipulation with Low-Cost Whole-Body Teleoperation [73]	2024	Yes	fine-tuned	not tested	not tested
On Bringing Robots Home [74]	2023	No	fine-tuned	not tested	not tested
What Matters in Language Conditioned Robotic Imitation Learning Over Unstructured Data [75]	2022	Yes	zero shot	not tested	not tested
BAKU: An Efficient Transformer for Multi-Task Policy Learning [76]	2024	Yes	not tested	not tested	not tested
Perceiver-Actor: A Multi-Task Transformer for Robotic Manipulation [77]	2022	Yes	not tested	not tested	not tested
RoboAgent: Generalization and Efficiency in Robot Manipulation via Semantic Augmentations and Action Chunking [78]	2024	Yes	zero shot	not tested	not tested
* Octo: An Open-Source Generalist Robot Policy [62]	2024	Yes	zero shot/fine-tuned	No	Yes
VIMA: General Robot Manipulation with Multimodal Prompts [79]	2023	Yes	zero shot	not tested	not tested
* Pushing the Limits of Cross-Embodiment Learning for Manipulation and Navigation [69]	2024	Yes	not tested	Yes	not tested

* Presented in detail below

Table 8.1: An analysis of approaches similar to RT-1-X shows that generalization to new embodiments is far less studied than generalization to new tasks, objects and environment, highlighting the relevance of this thesis in the field.

8.1.1 Pushing the limits of Cross-Embodiment Learning

This work follows a similar approach as RT-1-X by using a Transformer-based, end-to-end policy trained via imitation learning to investigate cross-embodiment transfer through the use of diverse datasets. However, the authors explicitly explore the extent of the embodiment gap that still allows for effective

knowledge transfer between different embodiments. While RT-1-X focuses primarily on manipulation tasks, utilizing almost exclusively robotic arms performing tabletop manipulation, this study broadens the scope by incorporating both navigation and manipulation datasets. For manipulation, the same subset of the Open X-Embodiment dataset used in RT-1-X is employed.

In contrast, the navigation data is drawn from a diverse set of datasets, including those involving mobile bases, quadrupeds, and even quadcopters. The training process is carefully balanced, with a 50/50 split between navigation and manipulation data.

One of the research questions particularly relevant to this thesis is: *Can heterogeneous cross-embodiment policies generalize zero-shot to new embodiments?* To address this, the policy is evaluated on the Mobile Aloha platform—a robotic arm mounted on a mobile base, which was not included in the training set. Remarkably, when tested on a simple pick-and-place task, the robot achieves a 50% success rate. This result is especially impressive given that both the embodiment and the environment were entirely unseen during training.

Several factors may contribute to the greater success in zero-shot generalization compared to RT-1-X. First, the authors took a more rigorous approach to aligning the action spaces across the different Open X-Embodiment datasets, manually verifying and adjusting them in a labor-intensive process. Additionally, rather than using language instructions to specify goals, they utilized goal images depicting the desired outcome. This likely simplifies zero-shot execution, as goal images inherently convey substantial information about the environment. However, the use of goal images is less convenient for human users, a limitation the authors themselves acknowledge.

While these factors may partially explain the improved zero-shot generalization to new embodiments, it is highly plausible that the broader variety of embodiments included in training, compared to RT-1-X, plays a significant role in facilitating transfer to an unseen embodiment. Nevertheless, further research is needed to determine the exact impact of these factors on zero-shot performance and to confirm whether this assumption holds true.

8.1.2 Octo

One of the most recent advancements in the field is Octo, introduced in May 2024 through a collaborative effort between UC Berkeley, Stanford, Carnegie Mellon University, and Google DeepMind. Many of the same researchers behind Open X-Embodiment and the RT models contributed to Octo's development. Similar to RT-1-X, Octo is a large, Transformer-based policy trained on a vast number of trajectories from the Open X-Embodiment dataset. However, Octo was specifically designed to facilitate transfer to new robotic setups through fine-tuning, and it has proven to be highly effective in this regard. Given its relevance to the aims of this thesis, Octo will be discussed in detail in the following section.

Octo shares several similarities with RT-1-X, but it introduces key innovations that enhance its adaptability. One of the main differences is the implementation of changeable readout heads on top of the Transformer, allowing the model to be adapted to new input or action modalities by only altering the readout head layer, rather than re-training large portions of the model. This feature makes it significantly easier to adapt Octo to robots with different action and observation spaces. Additionally, Octo supports goal specification through either language instructions or a goal image, offering greater flexibility in defining tasks. Octo is also trained on a much larger subset of the Open X-Embodiment dataset, benefiting from the expanded data available after RT-1-X was published. Consequently, Octo significantly outperforms RT-1-X in comparable scenarios.

One of Octo's primary advantages is that it is specifically designed with fine-tuning in mind. The fine-tuning process is optimized to run efficiently on consumer hardware, and the model's action heads allow it to be fine-tuned to robots with different action spaces. Furthermore, Octo can incorporate additional sensor data, such as force-torque inputs, making it more versatile. The authors demonstrated the model's ability to adapt to new embodiments using a dataset of just 100 demonstrations, achieving an average success rate of 72%, even in domains with new sensor inputs or action spaces.

Like RT-1-X, Octo's code is available as open-source software. However, the authors have provided a sophisticated, ready-to-use fine-tuning pipeline that is highly configurable. This pipeline addresses many of the challenges that had to be manually developed for this thesis. While it's unfortunate that Octo was released after this work began—meaning it could not be utilized within this thesis, which would have allowed more focus on evaluating performance rather than building and optimizing the fine-tuning process—it also highlights the relevance of the work done within this thesis. Additionally, it provides an opportunity for a meaningful comparison between the RT-1 and Octo architectures.

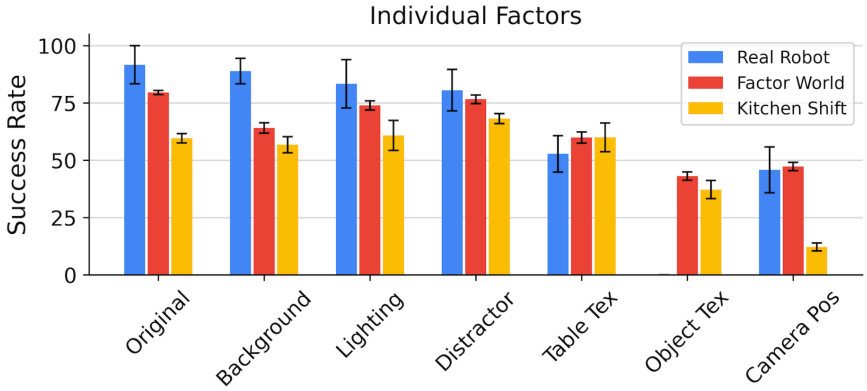

Figure 8.1: When decomposing the embodiment gap, it is found that changes in camera position and table texture have the highest impact on model performance. [80] This aligns with the findings of this thesis.

8.1.3 Exploring the Embodiment Gap

One of the key questions investigated in this thesis was the composition of the embodiment gap—the specific factors that influence a model's ability to generalize to new robotic embodiments. Through systematic experimentation, various factors were tested to determine their impact on model performance.

A recent study [80] addresses this question specifically: *What makes generalization hard for imitation learning in visual robotic manipulation?* Using the RT-1 model, the authors systematically alter environmental factors to assess how each change affects performance. Their findings, summarized in Figure 8.1, indicate that the factors with the most significant negative impact on performance are table texture and camera position.

These findings are particularly relevant to this thesis, as they align closely with the results observed here. In the experiments conducted, camera position was identified as having the most substantial influence on model behavior (Section 6.1), while changing the color of the table led to a dramatic decrease in pickup performance (Section 6.2.2). The consistency between the findings of this study and those of this thesis reinforce the validity of the results.

The authors also demonstrate that the performance decrease by such factors can be limited by techniques like data augmentation and pre-training on visual representations, techniques that are discussed in the following section.

8.2 Related Work in Different Directions

This section aims to explore research that is still in the area of generalization, following, however, slightly different approaches than RT-1-X and the works presented in the previous section, to give a broader overview of the domain.

Reinforcement Learning. A line of work investigates the same area of universal robotic learning as RT-1-X, but is based on reinforcement learning instead of imitation learning [81–86]. Many reinforcement learning approaches were found to perform well when fine-tuned on new embodiments, although no success was found zero-shot [87, 88].

Explicit transfer of knowledge. In contrast to simply fine-tuning pre-trained models on demonstrations from a new embodiment, some approaches tackle the embodiment gap more directly. Conditioning a model on the physical hardware properties of embodiments has shown to give the model more understanding of the differences in embodiment, allowing for successful transfer between robots [89, 90]. Additionally, learning the transfer of a policy from one domain to the other has also been proven effective [91, 92]. Another promising strategy involves combining multiple existing embodiment-specific policies into a single, unified policy capable of handling multiple robots [93].

Although not directly related to generalization between robotic embodiments, two other significant transfer challenges have been extensively explored in research:

- **Human-to-robot:** Transferring skills from videos of humans executing a task to robot policies. This approach is valuable due to the vast amount of such demonstrations available online, offering a rich resource compared to specific robotic datasets [94–99].
- **Sim-to-real:** Transferring skills learned in simulated environments to physical embodiments. This area has been thoroughly investigated because training in simulation is typically faster, cheaper, and safer than training in the real world [100–103].

Multiple Components. While this thesis focuses on end-to-end learning, other approaches concentrate on using models for specific components of the robotics pipeline, which are then combined to create a complete robotic control system. Some of these approaches demonstrate strong adaptability to new embodiments by modifying certain embodiment-specific components of the pipeline [104, 105]. Another line of research emphasizes learning generalizable visual representations, where fundamental scene understanding is shared between robots and can be used as a foundation for further fine-tuning [106–109]. Other approaches in this area explore chain-of-thought reasoning [110], navigation [18], and object detection [111].

Datasets. Open X-Embodiment presents the largest robotic demonstration dataset to date, as it combines many of the previously largest datasets into one resource [1]. It is widely used in various works within robotic manipulation research [62, 69].

However, one notable dataset not yet included in Open X-Embodiment is RH20T, which contains over 110,000 demonstrations of 150 different high-dexterity skills. The authors of RH20T emphasize high accuracy, employing specialized teleoperation setups with force feedback and calibrated camera systems. A unique feature of this dataset is the inclusion of corresponding human demonstrations—where a human manually performs the same task as the robot in the same environment—making it particularly valuable in human-to-robot learning. The exclusion of RH20T from Open X-Embodiment remains unexplained, and incorporating it could advance research in this area.

In addition to large-scale datasets, an interesting approach to increasing data diversity is artificial data augmentation. This technique involves expanding the dataset by manipulating images, such as adding distractors to improve environment generalization [78, 112], or even masking out the robot's embodiment in one dataset and replacing it with a target embodiment [113]. These methods offer a promising approach for enhancing generalization without the need for massive datasets.

8.3 Beyond RT-1-X

One of the most recent advancements in robotics research involves the development of multimodal models. These models go beyond traditional robotic data, leveraging knowledge from other domains, such as large language models [114, 115] and vision-language models [26]. As a result, these models are capable of performing tasks beyond robotic control, e.g. semantic reasoning [116] interactive dialogue [26].

Given that the RT-2 model [116] is a direct successor to the RT-1 model discussed in this thesis, it will be introduced in more detail here.

8.3.1 RT-2

Introduced by Google DeepMind in 2023, RT-2 builds on the foundation of RT-1, integrating a vision-language model trained on large-scale internet data into a robotic control system. The goal is to enhance generalization capabilities and enable additional semantic reasoning. The resulting model is called a vision-language-action model (VLA), as it extends the vision-language model with the ability to output robotic actions.

RT-2 leverages existing state-of-the-art models like PaLI-X and PaLM-E, which are capable of tasks such as image captioning and visual question answering. In RT-2, robotic actions are treated as text tokens and integrated into the training dataset alongside natural language tokens. This approach allows the model to be co-fine-tuned on both robotic trajectory data and large-scale vision-language tasks, enabling it to benefit from the generalization and semantic understanding already present in the underlying models.

While RT-2 performs similarly to RT-1 on seen tasks, it shows significant improvements when dealing with unseen objects, backgrounds, and environments. The inclusion of vision-language models leads to better generalization, as well as emergent capabilities, where the model can transfer semantic and visual

concepts from web data into robotic tasks. Notably, RT-2 excels in understanding symbols, reasoning, and human recognition, significantly outperforming RT-1 and other baselines across these categories.

These generalization abilities improve even further when pre-training RT-2 on the Open X-Embodiment dataset, resulting in the RT-2-X model [1]. Generalization of RT-2 or RT-2-X to new embodiments, zero-shot or fine-tuned, has however not been investigated yet, although the generalization properties to new environments makes this very interesting.

Currently it is not possible to investigate this as the code for RT-2 is not public, and it is unclear if it will be released in the future. Furthermore, due to its integration of a large vision-language model, RT-2 is significantly larger than RT-1 making training on consumer hardware impossible, large TPU clusters were used in training the model [1, 116]. Still, it would be very interesting if the generalization to new embodiments can also benefit from the incorporation of internet-scale data, and this will be an exciting future path of research.

Chapter 9

Conclusion

This thesis explored the generalization capabilities of the RT-1-X model, particularly its ability to adapt to new robotic embodiments. The research began with an evaluation of the model's zero-shot performance on the UMI robot, an embodiment not included in its pre-training. Based on the outcomes of these initial experiments, the focus then shifted to fine-tuning the model to enhance its performance on the UMI platform. Throughout the process, various factors influencing the embodiment gap were examined to gain a clearer understanding of the challenges involved in transferring robotic skills across different platforms. This chapter brings together the key findings from each stage of the research, discusses their implications for the field of robotic learning, and reflects on the insights gained. Finally, the chapter addresses the study's limitations and proposes directions for future work.

9.1 Key Findings

The research started with an in-depth analysis of the zero-shot performance of RT-1-X on the UMI robot, a new embodiment not included in the model's training data. The results clearly have demonstrated that RT-1-X is not capable of universal generalization to unseen embodiments without additional training. Despite extensive experimentation with various environmental setups, the model failed to exhibit any meaningful progress toward task completion in the zero-shot setting. This finding underscores the limitations of current end-to-end models in achieving true cross-embodiment transfer without any additional data.

Following the zero-shot experiments, the focus shifted to evaluating the model's adaptability through fine-tuning. A dataset of demonstrations was collected using the UMI robot, and a fine-tuning pipeline for RT-1-X was developed. The fine-tuning process effectively enhanced the model's performance on the new embodiment and the learned task, although accuracy issues were observed, and the performance did not reach the levels achieved on the embodiments seen during pre-training.

Additionally, it was found that no concrete knowledge about tasks and objects was transferred from the pre-training dataset. However, fine-tuning the pre-trained model still resulted in better performance and faster learning compared to training from scratch. This underscores the value of fine-tuning as a practical approach for adapting pre-trained models to specific robotic setups, despite its performance limitations.

9.2 Future work

9.2.1 Investigating Zero-Shot Transfer

RT-1-X has demonstrated the ability to transfer to new environments in a zero-shot manner, but this research has shown that it struggles to generalize to new embodiments in the same way. This raises important questions about the differences between embodiment transfer and environment transfer. An interesting direction for future research would be to completely decouple the embodiment from the environment. This could involve exactly replicating an environment from Open X-Embodiment (or ideally, conducting experiments in the original demonstration environment) while changing only the actual robot embodiment. This could even extend to only modifying specific components of the embodiment, such as the gripper or its color, to better isolate the effects.

To gain deeper insights into what the model perceives and whether it correctly identifies key parts of the image on the new embodiment (e.g. the end effector), it would be valuable to visualize the parts of the images the model attends to. Although this was beyond the scope of this thesis, as it would require modifications deep within the model architecture, adding this feature to the inference pipeline could be a valuable improvement.

9.2.2 Optimizing Fine-Tuning

This research has demonstrated that fine-tuning the RT-1-X model from Open X-Embodiment to the UMI robot is feasible, though not without limitations. This finding lays a solid foundation for further research aimed at enhancing the fine-tuning process.

Several approaches could be pursued: First, the fine-tuning pipeline itself could be optimized by adjusting hyperparameters, experimenting with training specific layers, or even incorporating new layers into the model.

Secondly, the application of the generalization apporaches discussed in the related works, such as policy transfer learning [91], could be applied to RT-1-X to investigate the effectiveness on the UMI robot.

Another approach for improvement could focus on increasing the efficiency of the fine-tuning process, particularly to achieve a larger batch size, which was a bottleneck in this research. Recent studies suggest that fine-tuning robotic transformer models can be made significantly more efficient [117].

Additionally, further experimentation could be conducted with the fine-tuning dataset itself: exploring the impact of fine-tuning on multiple tasks instead of just one, increasing the number of demonstrations, or applying data augmentation techniques [78, 112, 113] to enhance the fine-tuning dataset and potentially improve the model's generalization capabilities.

9.2.3 Other Models and Embodiments

This thesis focuses on adapting a single model to a single new embodiment. The literature study done within this thesis (Section 8.1) has shown that compared to generalization to new tasks and environments, generalization to new embodiments is much less explored. This presents an opportunity for future research to apply the same methodology used in this thesis to other target embodiments to determine if the results differ, or to experiment with different models to assess their impact on embodiment generalization.

The recently published Octo model [62] is particularly noteworthy in this context, as it is specifically optimized for fine-tuning. additionally, exploring embodiment generalization in vision-language-action (VLA) models like RT-2(-X) [1, 116] could be valuable, as these models leverage multimodal, internet-scale data, which may enhance their ability to generalize across different embodiments.

9.3 Final Remarks

In conclusion, this thesis has contributed to a better understanding of the challenges involved in generalizing robotic learning models, particularly in adapting pre-trained models like the RT-1-X to new robotic embodiments. The findings provide valuable insights that could inform future work in making robots more adaptable and efficient across different platforms. Although this research has highlighted some challenges, it also points to areas where further improvements can be made.

Acknowledgements

I would like to thank my supervisor, Arnoud Visser, for entrusting me with a topic that I thoroughly enjoyed working on, for providing me the freedom to explore it in my own way, and for introducing me to a field that I hope to remain involved in for a long time.

I also want to thank my friends and family for their invaluable help with proofreading, and for patiently listening to me compain about the ups and downs of my relationship with the robot.

Bibliography

[1] Open X-Embodiment Collaboration *et al.*, "Open X-Embodiment: Robotic Learning Datasets and RT-X Models," *arXiv.org*, Oct. 2023, ARXIV_ID: 2310.08864 MAG ID: 4387764390 S2ID: ef7d31137ef06c5be8c2824ecc5af6ce3358cc8f. DOI: 10.48550/arxiv.2310.08864.

[2] A. Brohan *et al.*, "RT-1: Robotics Transformer for Real-World Control at Scale," 2022, Publisher: [object Object] Version Number: 2. DOI: 10.48550/ARXIV.2212.06817. [Online]. Available: https://arxiv.org/abs/2212.06817 (visited on 04/09/2024).

[3] *Google-deepmind/open_x_embodiment*. [Online]. Available: https://github.com/google-deepmind/open_x_embodiment (visited on 08/29/2024).

[4] *Jonathansalzer/ROS2_rt-1-X*. [Online]. Available: https://github.com/jonathansalzer/ROS2_RT-1-X/tree/main (visited on 08/28/2024).

[5] J. Salzer, *Jonathansalzer/open_x_embodiment*, Jul. 2024. [Online]. Available: https://github.com/jonathansalzer/open_x_embodiment (visited on 08/28/2024).

[6] J. Salzer, *Jonathansalzer/LAB42_rtx_control*, Jun. 2024. [Online]. Available: https://github.com/jonathansalzer/LAB42_RTX_control (visited on 08/28/2024).

[7] J. Hua, L. Zeng, G. Li, and Z. Ju, "Learning for a Robot: Deep Reinforcement Learning, Imitation Learning, Transfer Learning," en, *Sensors*, vol. 21, no. 4, p. 1278, Jan. 2021, GSCC: 0000187 Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1424-8220. DOI: 10.3390/s21041278. [Online]. Available: https://www.mdpi.com/1424-8220/21/4/1278 (visited on 08/15/2024).

[8] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent Advances in Robot Learning from Demonstration," en, *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, May 2020, ISSN: 2573-5144, 2573-5144. DOI: 10.1146/annurev-control-100819-063206. [Online]. Available: https://www.annualreviews.org/doi/10.1146/annurev-control-100819-063206 (visited on 08/14/2024).

[9] S. Levine, C. Finn, T. Darrell, and P. Abbeel, *End-to-End Training of Deep Visuomotor Policies*, GSCC: 0004002 arXiv:1504.00702 [cs], Apr. 2016. DOI: 10.48550/arXiv.1504.00702. [Online]. Available: http://arxiv.org/abs/1504.00702 (visited on 08/21/2024).

[10] X. Xiao *et al.*, *Robot Learning in the Era of Foundation Models: A Survey*, GSCC: 0000206 arXiv:2311.14379 [cs], Nov. 2023. DOI: 10.48550/arXiv.2311.14379. [Online]. Available: http://arxiv.org/abs/2311.14379 (visited on 08/21/2024).

[11] R. Firoozi *et al.*, "Foundation Models in Robotics: Applications, Challenges, and the Future," 2023, Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2312.07843. [Online]. Available: https://arxiv.org/abs/2312.07843 (visited on 08/27/2024).

[12] Y. Hu *et al.*, *Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis*, arXiv:2312.08782 [cs], Dec. 2023. DOI: 10.48550/arXiv.2312.08782. [Online]. Available: http://arxiv.org/abs/2312.08782 (visited on 08/14/2024).

[13] D. Li *et al.*, *What Foundation Models can Bring for Robot Learning in Manipulation : A Survey*, GSCC: 0000067 arXiv:2404.18201 [cs], Aug. 2024. DOI: 10.48550/arXiv.2404.18201. [Online]. Available: http://arxiv.org/abs/2404.18201 (visited on 08/21/2024).

[14] *Google-research/robotics_transformer*, Aug. 2024. [Online]. Available: https://github.com/google-research/robotics_transformer (visited on 08/28/2024).

[15] A. Vaswani *et al.*, *Attention Is All You Need*, Version Number: 7, 2017. DOI: 10.48550/ARXIV.1706.03762. [Online]. Available: https://arxiv.org/abs/1706.03762 (visited on 07/01/2024).

[16] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, *Transformers in Vision: A Survey*, GSCC: 0002371 arXiv:2101.01169 [cs], Jan. 2022. DOI: 10.1145/3505244. [Online]. Available: http://arxiv.org/abs/2101.01169 (visited on 08/21/2024).

[17] Q. Wen *et al.*, *Transformers in Time Series: A Survey*, GSCC: 0000627 arXiv:2202.07125 [cs, eess, stat], May 2023. DOI: 10.48550/arXiv.2202.07125. [Online]. Available: http://arxiv.org/abs/2202.07125 (visited on 08/21/2024).

[18] A. Pashevich, C. Schmid, and C. Sun, *Episodic Transformer for Vision-and-Language Navigation*, GSCC: 0000184 arXiv:2105.06453 [cs], Aug. 2021. DOI: 10.48550/arXiv.2105.06453. [Online]. Available: http://arxiv.org/abs/2105.06453 (visited on 08/21/2024).

[19] Michael Ahn *et al.*, "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances," *Conference on Robot Learning*, 2022.

[20] Eric Jang *et al.*, "BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning," *Conference on Robot Learning*, 2022, ARXIV_ID: 2202.02005 S2ID: 1d803f07e4591bd67c358eef715bcd443e821894.

[21] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," 2019, Publisher: arXiv Version Number: 5. DOI: 10.48550/ARXIV.1905.11946. [Online]. Available: https://arxiv.org/abs/1905.11946 (visited on 07/01/2024).

[22] O. Russakovsky *et al.*, *ImageNet Large Scale Visual Recognition Challenge*, Version Number: 3, 2014. DOI: 10.48550/ARXIV.1409.0575. [Online]. Available: https://arxiv.org/abs/1409.0575 (visited on 07/01/2024).

[23] D. Cer *et al.*, *Universal Sentence Encoder*, Version Number: 2, 2018. DOI: 10.48550/ARXIV.1803.11175. [Online]. Available: https://arxiv.org/abs/1803.11175 (visited on 07/01/2024).

[24] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "FiLM: Visual Reasoning with a General Conditioning Layer," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018, ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v32i1.11671. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11671 (visited on 07/01/2024).

[25] M. S. Ryoo, A. Piergiovanni, A. Arnab, M. Dehghani, and A. Angelova, "TokenLearner: What Can 8 Learned Tokens Do for Images and Videos?," 2021, Publisher: arXiv Version Number: 4. DOI: 10.48550/ARXIV.2106.11297. [Online]. Available: https://arxiv.org/abs/2106.11297 (visited on 07/01/2024).

[26] S. Reed *et al.*, *A Generalist Agent*, arXiv:2205.06175 [cs], Nov. 2022. DOI: 10.48550/arXiv.2205.06175. [Online]. Available: http://arxiv.org/abs/2205.06175 (visited on 07/08/2024).

[27] K.-H. Lee *et al.*, *Multi-Game Decision Transformers*, arXiv:2205.15241 [cs], Oct. 2022. DOI: 10.48550/arXiv.2205.15241. [Online]. Available: http://arxiv.org/abs/2205.15241 (visited on 08/28/2024).

[28] E. Jang *et al.*, *BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning*, GSCC: 0000375 arXiv:2202.02005 [cs], Feb. 2022. DOI: 10.48550/arXiv.2202.02005. [Online]. Available: http://arxiv.org/abs/2202.02005 (visited on 08/16/2024).

[29] *Personal Robotics Seminar, London 1985 - Developing UMI's R Theta mobile robot.* [Online]. Available: http://davidbuckley.net/RS/History/LondonRobotics85/TimJonesUMI.htm (visited on 07/03/2024).

[30] T. Jones, *UMI R-Theta*, Mar. 2021. [Online]. Available: https://web.archive.org/web/20240704092454/https://www.linkedin.com/posts/tim-jones-8464751_in-1983-geoff-henny-i-set-up-universal-activity-6780562127376674816-D5nA (visited on 04/07/2024).

[31] T. Jones, *UMI RTX*, Mar. 2021. [Online]. Available: https://web.archive.org/web/20240704093612/https://www.linkedin.com/posts/tim-jones-8464751_in-1984-we-recognised-that-there-wasnt-an-activity-6782593664372772864-VYJq/ (visited on 04/07/2024).

[32] H. Bassily, R. Sekhon, D. E. Butts, and J. Wagner, "A mechatronics educational laboratory – Programmable logic controllers and material handling experiments," en, *Mechatronics*, vol. 17, no. 9, pp. 480–488, Nov. 2007, ISSN: 09574158. DOI: 10.1016/j.mechatronics.2007.06.004. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957415807000670 (visited on 07/03/2024).

[33] J. Hollingum, "Intelligent machines from Oxford," en, *Industrial Robot: An International Journal*, vol. 21, no. 5, pp. 35–37, Oct. 1994, ISSN: 0143-991X. DOI: 10.1108/EUM0000000004165. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/EUM0000000004165/full/html (visited on 07/03/2024).

[34] H. M. Van Der Loos and D. J. Reinkensmeyer, "Rehabilitation and Health Care Robotics," en, in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1223–1251, ISBN: 978-3-540-23957-4 978-3-540-30301-5. DOI: 10.1007/978-3-540-30301-5_54. [Online]. Available: http://link.springer.com/10.1007/978-3-540-30301-5_54 (visited on 07/03/2024).

[35] N. Brown, *Advanced Vision Module - TASK 5 MATLAB CODE*, Publisher: School of Informatics, The University of Edinburgh. [Online]. Available: https://www.inf.ed.ac.uk/teaching/courses/av/MATLAB/TASK5/ (visited on 07/04/2024).

[36] P. Bauters, R. De Keyser, Y. Song, and P. Van Renterghem, "A Transputer-Based Controller for the RTX Robot," en, *IFAC Proceedings Volumes*, vol. 25, no. 20, pp. 61–65, Sep. 1992, ISSN: 14746670. DOI: 10.1016/S1474-6670(17)49839-9. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1474667017498399 (visited on 07/03/2024).

[37] K. K. Chintamani, A. K. Nawab, A. K. Pandy, R. D. Ellis, A. Cao, and G. A. Cao, "Comparing Two Kinematics Methods for Telerobotic Control Applications," en, *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 50, no. 1, pp. 126–130, Oct. 2006, ISSN: 2169-5067, 1071-1813. DOI: 10.1177/154193120605000127. [Online]. Available: http://journals.sagepub.com/doi/10.1177/154193120605000127 (visited on 07/03/2024).

[38] M. Fewless, G. Slater, and S. Waldron, "Telerobotic Controller," en,

[39] S. Prior and P. Warner, "A review of world rehabilitation robotics research," in *IEE Colloquium on High-Tech Help for the Handicapped*, Apr. 1990, pp. 1/1–1/3. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/189943 (visited on 07/04/2024).

[40] M. P. Dijkers, P. C. deBear, R. F. Erlandson, K. Kristy, D. M. Geer, and A. Nichols, "Patient and staff acceptance of robotic technology in occupational therapy: A pilot study," eng, *Journal of Rehabilitation Research and Development*, vol. 28, no. 2, pp. 33–44, 1991, ISSN: 0748-7711. DOI: 10.1682/jrrd.1991.04.0033.

[41] M. Busnel, R. Cammoun, F. Coulon-Lauture, J. M. Détriché, G. Le Claire, and B. Lesigne, "The robotized workstation "MASTER" for users with tetraplegia: Description and evaluation," eng, *Journal of Rehabilitation Research and Development*, vol. 36, no. 3, pp. 217–229, Jul. 1999, ISSN: 0748-7711.

[42] C. Fu, "An Independent Vocational Workstation for a Quadriplegic," *Interactive Robotic Aids–One Option for Independent Living: An International Perspective.*, vol. Monograph Number 37, 1986, ISBN-939986-50-1. [Online]. Available: https://files.eric.ed.gov/fulltext/ED285374.pdf.

[43] J. Hammel *et al.*, "Clinical evaluation of a desktop robotic assistant," *Journal of rehabilitation research and development*, vol. 26, pp. 1–16, Feb. 1989.

[44] J. Schuyler and R. Mahoney, "Assessing human-robotic performance for vocational placement," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 3, pp. 394–404, Sep. 2000, ISSN: 10636528. DOI: 10.1109/86.867881. [Online]. Available: http://ieeexplore.ieee.org/document/867881/ (visited on 07/03/2024).

[45] S. Effendi, R. Jarvis, and D. Suter, "Robot Manipulation Grasping of Recognized Objects for Assistive Technology Support Using Stereo Vision," en, 2008.

[46] R. G. Gosine, W. S. Harwin, L. J. Furby, and R. D. Jackson, "An intelligent end-effector for a rehabilitation robot," en, *Journal of Medical Engineering & Technology*, vol. 13, no. 1-2, pp. 37–43, Jan. 1989, ISSN: 0309-1902, 1464-522X. DOI: 10.3109/03091908909030192. [Online]. Available: http://www.tandfonline.com/doi/full/10.3109/03091908909030192 (visited on 07/03/2024).

[47] M. R. Hillman, "Design and development of a robotic workstation for the disabled," PhD Thesis, University of Bath, Bath, 1992.

[48] T. Jones, *UMI Fish Robot*, Feb. 2021. [Online]. Available: https://web.archive.org/web/20240704151444/https://www.linkedin.com/posts/tim-jones-8464751_another-archive-item-from-the-90s-fish-activity-6805812504095150080-eEOQ (visited on 04/07/2024).

[49] X. Dooms, "Camera gebaseerde robotstu- ring d.m.v. ROS implementatie met OpenCV," nl, M.S. thesis, KU Leuven, 2015.

[50] T. D. Nayer, "Camera gebaseerde robotsturing," nl, M.S. thesis, KU Leuven, 2016.

[51] F. Groen, G. der Boer, A. van Inge, and R. Stam, "A chess playing robot: Lab course in robot sensor integration," in *[1992] Conference Record IEEE Instrumentation and Measurement Technology Conference*, May 1992, pp. 261–264. DOI: 10.1109/IMTC.1992.245137. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/245137 (visited on 07/04/2024).

[52] N. Peek and A. Nusselder, *UMI RTX plotter*. [Online]. Available: https://staff.fnwi.uva.nl/a.visser/education/ZSB/2006/Experiment/AmbientPlotter/with/ (visited on 07/04/2024).

[53] A. Visser, *UMI RTX control*, original-date: 2022-02-24T14:12:55Z, May 2024. [Online]. Available: https://github.com/physar/umi-rtx (visited on 07/04/2024).

[54] G. Garde and T. Massa, "A ROS 2 Interface for the UMI-RTX robotic arm," Internship Report, ENSTA Bretagne and the University of Amsterdam, 2023.

[55] G. Garde and T. Massa, *UMI ROS2 interface*, original-date: 2023-05-04T07:54:51Z, May 2024. [Online]. Available: https://github.com/gardegu/LAB42_RTX_control (visited on 07/04/2024).

[56] U. M. I. Ltd, *Inside RTX*. Aug. 1987.

[57] J. Carpentier *et al.*, "The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," en, Jan. 2019. [Online]. Available: https://laas.hal.science/hal-01866228 (visited on 08/28/2024).

[58] *MaintenanceManualForRTX*. [Online]. Available: https://staff.science.uva.nl/a.visser/education/ZSB/MaintenanceManualForRTX.pdf (visited on 08/28/2024).

[59] H. Walke *et al.*, *BridgeData V2: A Dataset for Robot Learning at Scale*, Version Number: 3, 2023. DOI: 10.48550/ARXIV.2308.12952. [Online]. Available: https://arxiv.org/abs/2308.12952 (visited on 07/03/2024).

[60] G. Paaß and S. Giesselbach, *Foundation Models for Natural Language Processing – Pre-trained Language Models Integrating Media*, Version Number: 1, 2023. DOI: 10.48550/ARXIV.2302.08575. [Online]. Available: https://arxiv.org/abs/2302.08575 (visited on 07/14/2024).

[61] J. Howard and S. Ruder, *Universal Language Model Fine-tuning for Text Classification*, arXiv:1801.06146 [cs, stat], May 2018. DOI: 10.48550/arXiv.1801.06146. [Online]. Available: http://arxiv.org/abs/1801.06146 (visited on 07/08/2024).

[62] Octo Model Team *et al.*, "Octo: An Open-Source Generalist Robot Policy," *arXiv.org*, 2024, GSCC: 0000053 ARXIV_ID: 2405.12213 S2ID: 1d2753d74025e7a71594506623be81f18b073adb. DOI: 10.48550/arxiv.2405.12213.

[63] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [cs], Jan. 2017. DOI: 10.48550/arXiv.1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980 (visited on 07/29/2024).

[64] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, arXiv:1810.04805 [cs], May 2019. DOI: 10.48550/arXiv.1810.04805. [Online]. Available: http://arxiv.org/abs/1810.04805 (visited on 07/29/2024).

[65] *Open Source Survey*, en. [Online]. Available: https://opensourcesurvey.org/2017/ (visited on 08/16/2024).

[66] S. Sonnenburg *et al.*, "The Need for Open Source Software in Machine Learning," en, GSCC: 0000277.

[67] L. Bahaidarah, E. Hung, A. F. D. M. Oliveira, J. Penumaka, L. Rosario, and A. Trisovic, *Toward Reusable Science with Readable Code and Reproducibility*, en, GSCC: 0000002 arXiv:2109.10387 [cs], Sep. 2021. [Online]. Available: http://arxiv.org/abs/2109.10387 (visited on 08/16/2024).

[68] B. Haibe-Kains *et al.*, "Transparency and reproducibility in artificial intelligence," en, *Nature*, vol. 586, no. 7829, E14–E16, Oct. 2020, Publisher: Nature Publishing Group, ISSN: 1476-4687. DOI: 10.1038/s41586-020-2766-y. [Online]. Available: https://www.nature.com/articles/s41586-020-2766-y (visited on 08/28/2024).

[69] J. Yang *et al.*, "Pushing the Limits of Cross-Embodiment Learning for Manipulation and Navigation," 2024, GSCC: 0000005 Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2402.19432. [Online]. Available: https://arxiv.org/abs/2402.19432 (visited on 08/27/2024).

[70] K. Kawaharazuka, T. Matsushima, A. Gambardella, J. Guo, C. Paxton, and A. Zeng, *Real-World Robot Applications of Foundation Models: A Review*, arXiv:2402.05741 [cs], Feb. 2024. [Online]. Available: http://arxiv.org/abs/2402.05741 (visited on 04/10/2024).

[71] Jonathan T. Yang, Dorsa Sadigh, and Chelsea Finn, "Polybot: Training One Policy Across Robots While Embracing Variability," *Conference on Robot Learning*, Jul. 2023, GSCC: 0000011 ARXIV_ID: 2307.03719 MAG ID: 4383860855 S2ID: e46aa32130e6df1d7055d51fd3e5782a22dae2a1. DOI: 10.48550/arxiv.2307.03719.

[72] I. Radosavovic, B. Shi, L. Fu, K. Goldberg, T. Darrell, and J. Malik, "Robot Learning with Sensorimotor Pre-training," 2023, GSCC: 0000025 Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2306.10007. [Online]. Available: https://arxiv.org/abs/2306.10007 (visited on 08/27/2024).

[73] Zipeng Fu, Tony Zhao, and Chelsea Finn, "Mobile ALOHA: Learning Bimanual Mobile Manipulation with Low-Cost Whole-Body Teleoperation," *arXiv.org*, 2024, GSCC: 0000093 ARXIV_ID: 2401.02117 S2ID: fc3819a50705fc3cf90ab92f2a206b858fef3b19. DOI: 10.48550/arxiv.2401.02117.

[74] N. M. M. Shafiullah *et al.*, *On Bringing Robots Home*, en, Nov. 2023. [Online]. Available: https://arxiv.org/abs/2311.16098v1 (visited on 08/14/2024).

[75] O. Mees, L. Hermann, and W. Burgard, "What Matters in Language Conditioned Robotic Imitation Learning Over Unstructured Data," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 205–11 212, Oct. 2022, ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2022.3196123. [Online]. Available: https://ieeexplore.ieee.org/document/9849097/ (visited on 08/27/2024).

[76] S. Haldar, Z. Peng, and L. Pinto, "BAKU: An Efficient Transformer for Multi-Task Policy Learning," 2024, GSCC: 0000000 Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2406.07539. [Online]. Available: https://arxiv.org/abs/2406.07539 (visited on 08/27/2024).

[77] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-Actor: A Multi-Task Transformer for Robotic Manipulation," GSCC: 0000330 Version Number: 2, arXiv, 2022. DOI: 10.48550/ARXIV.2209.05451. [Online]. Available: https://arxiv.org/abs/2209.05451 (visited on 08/27/2024).

[78] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar, "RoboAgent: Generalization and Efficiency in Robot Manipulation via Semantic Augmentations and Action Chunking," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4788–4795, May 2024, GSCC: 0000039 Conference Name: 2024 IEEE International Conference on Robotics and Automation (ICRA) ISBN: 9798350384574 Place: Yokohama, Japan Publisher: IEEE. DOI: 10.1109/ICRA57147.2024.10611293. [Online]. Available: https://ieeexplore.ieee.org/document/10611293/ (visited on 08/27/2024).

[79] Y. Jiang *et al.*, *VIMA: General Robot Manipulation with Multimodal Prompts*, GSCC: 0000153 arXiv:2210.03094 [cs], May 2023. DOI: 10.48550/arXiv.2210.03094. [Online]. Available: http://arxiv.org/abs/2210.03094 (visited on 08/16/2024).

[80] A. Xie, L. Lee, T. Xiao, and C. Finn, "Decomposing the Generalization Gap in Imitation Learning for Visual Robotic Manipulation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, GSCC: 0000016, May 2024, pp. 3153–3160. DOI: 10.1109/ICRA57147.2024.10611331. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10611331 (visited on 08/14/2024).

[81] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2169–2176, May 2017, GSCC: 0000436 Conference Name: 2017 IEEE International Conference on Robotics and Automation (ICRA) ISBN: 9781509046331 Place: Singapore, Singapore Publisher: IEEE. DOI: 10.1109/ICRA.2017.7989250. [Online]. Available: http://ieeexplore.ieee.org/document/7989250/ (visited on 08/27/2024).

[82] R. Rafailov, K. B. Hatch, V. Kolev, J. D. Martin, M. Phielipp, and C. Finn, "MOTO: Offline Pre-training to Online Fine-tuning for Model-based Robot Learning," en, in *Proceedings of The 7th Conference on Robot Learning*, GSCC: 0000008 ISSN: 2640-3498, PMLR, Dec. 2023, pp. 3654–3671. [Online]. Available: https://proceedings.mlr.press/v229/rafailov23a.html (visited on 08/14/2024).

[83] R. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman, *Never Stop Learning: The Effectiveness of Fine-Tuning in Robotic Reinforcement Learning*, GSCC: 0000071 arXiv:2004.10190 [cs, stat], Jul. 2020. DOI: 10.48550/arXiv.2004.10190. [Online]. Available: http://arxiv.org/abs/2004.10190 (visited on 08/14/2024).

[84] J. Yang, M. S. Mark, B. Vu, A. Sharma, J. Bohg, and C. Finn, "Robot Fine-Tuning Made Easy: Pre-Training Rewards and Policies for Autonomous Real-World Reinforcement Learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, GSCC: 0000006, May 2024, pp. 4804–4811. DOI: 10.1109/ICRA57147.2024.10610421. [Online]. Available: https://ieeexplore.ieee.org/document/10610421 (visited on 08/14/2024).

[85] D. Kalashnikov *et al.*, "MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale," *ArXiv*, Apr. 2021. [Online]. Available: https://www.semanticscholar.org/paper/MT-Opt%3A-Continuous-Multi-Task-Robotic-Reinforcement-Kalashnikov-Varley/3e85d208b1b927fdb69ecf8336c70995818aaebd?sort=relevance&queryString=open%20x (visited on 08/27/2024).

[86] K.-H. Lee, T. Xiao, A. Li, P. Wohlhart, I. Fischer, and Y. Lu, *PI-QT-Opt: Predictive Information Improves Multi-Task Robotic Reinforcement Learning at Scale*, arXiv:2210.08217 [cs, math], Nov. 2022. DOI: 10.48550/arXiv.2210.08217. [Online]. Available: http://arxiv.org/abs/2210.08217 (visited on 07/08/2024).

[87] S. Dasari *et al.*, "RoboNet: Large-Scale Multi-Robot Learning," *ArXiv*, Oct. 2019, GSCC: 0000282. [Online]. Available: https://www.semanticscholar.org/paper/RoboNet%3A-Large-Scale-Multi-Robot-Learning-Dasari-Ebert/3e519d85cdcefdd1d2ad89829d6ad445695d8c58 (visited on 08/27/2024).

[88] K. Bousmalis *et al.*, *RoboCat: A Self-Improving Generalist Agent for Robotic Manipulation*, GSCC: 0000010 arXiv:2306.11706 [cs], Dec. 2023. DOI: 10.48550/arXiv.2306.11706. [Online]. Available: http://arxiv.org/abs/2306.11706 (visited on 08/14/2024).

[89] A. Gupta, L. Fan, S. Ganguli, and L. Fei-Fei, "MetaMorph: Learning Universal Controllers with Transformers," 2022, GSCC: 0000070 Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2203.11931. [Online]. Available: https://arxiv.org/abs/2203.11931 (visited on 08/27/2024).

[90] T. Chen, A. Murali, and A. Gupta, *Hardware Conditioned Policies for Multi-Robot Transfer Learning*, GSCC: 0000091 arXiv:1811.09864 [cs], Jan. 2019. DOI: 10.48550/arXiv.1811.09864. [Online]. Available: http://arxiv.org/abs/1811.09864 (visited on 08/14/2024).

[91] G. Zhang, L. Zhong, Y. Lee, and J. J. Lim, *Policy Transfer across Visual and Dynamics Domain Gaps via Iterative Grounding*, GSCC: 0000010 arXiv:2107.00339 [cs], Jul. 2021. DOI: 10.48550/arXiv.2107.00339. [Online]. Available: http://arxiv.org/abs/2107.00339 (visited on 08/27/2024).

[92] A. Ghadirzadeh, X. Chen, P. Poklukar, C. Finn, M. Björkman, and D. Kragic, *Bayesian Meta-Learning for Few-Shot Policy Adaptation Across Robotic Platforms*, GSCC: 0000031 arXiv:2103.03697 [cs], Mar. 2021. DOI: 10.48550/arXiv.2103.03697. [Online]. Available: http://arxiv.org/abs/2103.03697 (visited on 08/14/2024).

[93] L. Wang, J. Zhao, Y. Du, E. H. Adelson, and R. Tedrake, "PoCo: Policy Composition from and for Heterogeneous Robot Learning," 2024, GSCC: 0000004 Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2402.02511. [Online]. Available: https://arxiv.org/abs/2402.02511 (visited on 08/27/2024).

[94] D. Guo, "Learning Multi-Step Manipulation Tasks from A Single Human Demonstration," 2023, GSCC: 0000003 Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2312.15346. [Online]. Available: https://arxiv.org/abs/2312.15346 (visited on 08/27/2024).

[95] A. Mandlekar *et al.*, "MimicGen: A Data Generation System for Scalable Robot Learning using Human Demonstrations," GSCC: 0000030 Version Number: 1, arXiv, 2023. DOI: 10.48550/ARXIV.2310.17596. [Online]. Available: https://arxiv.org/abs/2310.17596 (visited on 08/27/2024).

[96] J. Zhou, T. Ma, K.-Y. Lin, R. Qiu, Z. Wang, and J. Liang, "Mitigating the Human-Robot Domain Discrepancy in Visual Pre-training for Robotic Manipulation," 2024, GSCC: 0000001 Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2406.14235. [Online]. Available: https://arxiv.org/abs/2406.14235 (visited on 08/27/2024).

[97] P. Sharma, L. Mohan, L. Pinto, and A. Gupta, "Multiple Interactions Made Easy (MIME): Large Scale Demonstrations Data for Imitation," GSCC: 0000103, Oct. 2018. [Online]. Available: https://www.semanticscholar.org/paper/Multiple-Interactions-Made-Easy-(MIME)%3A-Large-Scale-Sharma-Mohan/df350766b53d4db23790a0408b0d2c7a185cff74 (visited on 08/27/2024).

[98] K. Zakka, A. Zeng, P. R. Florence, J. Tompson, J. Bohg, and D. Dwibedi, "XIRL: Cross-embodiment Inverse Reinforcement Learning," GSCC: 0000099, Jun. 2021. [Online]. Available: https://www.semanticscholar.org/paper/XIRL%3A-Cross-embodiment-Inverse-Reinforcement-Zakka-Zeng/61ff6872fa07788acdbe85b1319554a5fb9ce0db (visited on 08/27/2024).

[99] T. He *et al.*, "Learning Human-to-Humanoid Real-Time Whole-Body Teleoperation," 2024, Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2403.04436. [Online]. Available: https://arxiv.org/abs/2403.04436 (visited on 08/28/2024).

[100] Y. Chebotar *et al.*, "Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8973–8979, May 2019, Conference Name: 2019 International Conference on Robotics and Automation (ICRA) ISBN: 9781538660270 Place: Montreal, QC, Canada Publisher: IEEE. DOI: 10.1109/ICRA.2019.8793789. [Online]. Available: https://ieeexplore.ieee.org/document/8793789/ (visited on 08/28/2024).

[101] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3803–3810, May 2018, Conference Name: 2018 IEEE International Conference on Robotics and Automation (ICRA) ISBN: 9781538630815 Place: Brisbane, QLD Publisher: IEEE. DOI: 10.1109/ICRA.2018.8460528. [Online]. Available: https://ieeexplore.ieee.org/document/8460528/ (visited on 08/28/2024).

[102] A. Handa *et al.*, "DeXtreme: Transfer of Agile In-hand Manipulation from Simulation to Reality," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5977–5984, May 2023, Conference Name: 2023 IEEE International Conference on Robotics and Automation (ICRA) ISBN: 9798350323658 Place: London, United Kingdom Publisher: IEEE. DOI: 10.1109/ICRA48891.2023.10160216. [Online]. Available: https://ieeexplore.ieee.org/document/10160216/ (visited on 08/28/2024).

[103] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei, *TRANSIC: Sim-to-Real Policy Transfer by Learning from Online Correction*, en, arXiv:2405.10315 [cs], May 2024. [Online]. Available: http://arxiv.org/abs/2405.10315 (visited on 08/28/2024).

[104] Peiqi Liu, Yaswanth Orru, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto, "OK-Robot: What Really Matters in Integrating Open-Knowledge Models for Robotics," *arXiv.org*, 2024, GSCC: 0000025 ARXIV_ID: 2401.12202 S2ID: 7220c698ea7e42528a39e80405947fbacd72fbbf. DOI: 10.48550/arxiv.2401.12202.

[105] Jiaqiang Ye Zhu, Carla Gomez Cano, David Vazquez Bermudez, and Michal Drozdzal, "InCoRo: In-Context Learning for Robotics Control with Feedback Loops," *arXiv.org*, 2024, GSCC: 0000003 ARXIV_ID: 2402.05188 S2ID: 7ceefa6766418966a12a04d282967258e5cd1267. DOI: 10.48550/arxiv.2402.05188.

[106] J. Pari, N. M. Shafiullah, S. P. Arunachalam, and L. Pinto, *The Surprising Effectiveness of Representation Learning for Visual Imitation*, GSCC: 0000108 arXiv:2112.01511 [cs], Dec. 2021. DOI: 10.48550/arXiv.2112.01511. [Online]. Available: http://arxiv.org/abs/2112.01511 (visited on 08/13/2024).

[107] Suraj Nair, A. Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhi Gupta, "R3M: A Universal Visual Representation for Robot Manipulation," *Conference on Robot Learning*, 2022, GSCC: 0000379 ARXIV_ID: 2203.12601 S2ID: c9bdc9ad2c3cf3230ba9aac7b5783ab411f0d204. DOI: 10.48550/arxiv.2203.12601.

[108] S. Chen, R. Garcia, I. Laptev, and C. Schmid, *SUGAR: Pre-training 3D Visual Representations for Robotics*, arXiv:2404.01491 [cs], Apr. 2024. DOI: 10.48550/arXiv.2404.01491. [Online]. Available: http://arxiv.org/abs/2404.01491 (visited on 08/28/2024).

[109] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell, *Real-World Robot Learning with Masked Visual Pre-training*, GSCC: 0000184 arXiv:2210.03109 [cs], Oct. 2022. DOI: 10.48550/arXiv.2210.03109. [Online]. Available: http://arxiv.org/abs/2210.03109 (visited on 08/14/2024).

[110] Y. Mu *et al.*, "EmbodiedGPT: Vision-Language Pre-Training via Embodied Chain of Thought," 2023, GSCC: 0000121 Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2305.15021. [Online]. Available: https://arxiv.org/abs/2305.15021 (visited on 08/27/2024).

[111] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu, *VIOLA: Imitation Learning for Vision-Based Manipulation with Object Proposal Priors*, arXiv:2210.11339 [cs], Mar. 2023. DOI: 10.48550/arXiv.2210.11339. [Online]. Available: http://arxiv.org/abs/2210.11339 (visited on 08/28/2024).

[112] T. Yu *et al.*, "Scaling Robot Learning with Semantically Imagined Experience," 2023, GSCC: 0000076 Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2302.11550. [Online]. Available: https://arxiv.org/abs/2302.11550 (visited on 08/27/2024).

[113] L. Y. Chen, K. Hari, K. Dharmarajan, C. Xu, Q. Vuong, and K. Goldberg, "Mirage: Cross-Embodiment Zero-Shot Policy Transfer with Cross-Painting," 2024, GSCC: 0000002 Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2402.19249. [Online]. Available: https://arxiv.org/abs/2402.19249 (visited on 08/27/2024).

[114] Danny Driess *et al.*, "PaLM-E: An Embodied Multimodal Language Model," *International Conference on Machine Learning*, Mar. 2023, GSCC: 0001104 ARXIV_ID: 2303.03378 MAG ID: 4323572061 S2ID: 38fe8f324d2162e63a967a9ac6648974fc4c66f3. DOI: 10.48550/arxiv.2303.03378.

[115] F. Zeng, W. Gan, Y. Wang, N. Liu, and P. S. Yu, "Large Language Models for Robotics: A Survey," 2023, Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2311.07226. [Online]. Available: https://arxiv.org/abs/2311.07226 (visited on 08/27/2024).

[116] A. Brohan *et al.*, "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," 2023, GSCC: 0000446. DOI: 10.48550/ARXIV.2307.15818. [Online]. Available: https://arxiv.org/abs/2307.15818 (visited on 01/11/2024).

[117] I. Leal *et al.*, *SARA-RT: Scaling up Robotics Transformers with Self-Adaptive Robust Attention*, GSCC: 0000004 arXiv:2312.01990 [cs], Dec. 2023. DOI: 10.48550/arXiv.2312.01990. [Online]. Available: http://arxiv.org/abs/2312.01990 (visited on 08/16/2024).

[118] A. Naceri *et al.*, "The Vicarios Virtual Reality Interface for Remote Robotic Teleoperation: Teleporting for Intuitive Tele-manipulation," en, *Journal of Intelligent & Robotic Systems*, vol. 101, no. 4, p. 80, Apr. 2021, ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-021-01311-7. [Online]. Available: https://link.springer.com/10.1007/s10846-021-01311-7 (visited on 07/18/2024).

[119] J. I. Lipton, A. J. Fay, and D. Rus, "Baxter's Homunculus: Virtual Reality Spaces for Teleoperation in Manufacturing," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 179–186, Jan. 2018, ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2017.2737046. [Online]. Available: http://ieeexplore.ieee.org/document/8003431/ (visited on 07/18/2024).

[120] C. Melchiorri, "Robot Teleoperation," en, in *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Eds., London: Springer London, 2013, pp. 1–14, ISBN: 978-1-4471-5102-9. DOI: 10.1007/978-1-4471-5102-9_172-1. [Online]. Available: https://link.springer.com/10.1007/978-1-4471-5102-9_172-1 (visited on 07/18/2024).

[121] G. Niemeyer, C. Preusche, S. Stramigioli, and D. Lee, "Telerobotics," en, in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., Series Title: Springer Handbooks, Cham: Springer International Publishing, 2016, pp. 1085–1108, ISBN: 978-3-319-32550-7 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1_43. [Online]. Available: https://link.springer.com/10.1007/978-3-319-32552-1_43 (visited on 07/18/2024).

Appendix A

Integration

A.1 Action Scaling

Table A.1 shows the determined minimal and maximal coordinate values for the UMI robot, along with the respective value range of the RT-1-X model.

Axis	UMI min	UMI max	UMI range	RT-1-X min	RT-1-X max
x	-0.5	0.5	1	-2	2
y	0.2	0.7	0.5	-2	2
z	0.2	0.7	0.5	-2	2
Yaw (rot x)	45	135	90	$-\pi/2$	$\pi/2$
Pitch (rot y)	0	90	90	$-\pi/2$	$\pi/2$
Roll (rot z)	0	90	90	$-\pi/2$	$\pi/2$
Grip	0.02 (fully closed)	0.08 (fully open)	0.06	1 (fully closed)	-1 (fully open)

Table A.1: Actions need to be scaled from the model range to the range of movement of the UMI robot.

Appendix B

Experiments

B.1 Selection of the Teleoperation Strategy

Two qualities are important when choosing an approach for this task: It needs to be accurate enough to execute pick-and-place tasks reliably, and it needs to be efficient, to allow the collection of a large amount of demonstrations in a suitable timeframe. For those reasons,

When looking at the datasets in Open X-Embodiments, 13 different methods of robot control are used, see Figure B.1. The majority of demonstrations are collected via human virtual reality control and similar approaches. VR has established itself as a well suited mode for robotic teleoperation due to the immersive experience it provides to the human operator [118, 119]. It is however not trivial to implement, requiring specific hardware and software, the setup of which would go beyond the scope of this thesis.

Furthermore, the UMI robot comes with the additional challenge of having a significant delay between receiving a command, and executing the respective motor movements. This control problem is well known, although it does not have much relevancy in today's robotics landscape, as robots have become much faster and are able to operate in de-facto real time [120, 121]. Due to its age, the UMI still suffers from this issue, and this delay in inputs renders the advantages of VR teleoperation useless, as the UMI movements would be significantly slower than the human operator.

There are some datasets in Open X-Embodiment that use simpler methods of data collection, e.g. a joystick or keyboard. When evaluating those options, it was decided to use a gamepad in the style of a PlayStation controller for gathering demonstrations on the UMI: It features two joystick-style inputs as well as a variety of buttons, combining the advantages of joystick and keyboard control, the joysticks enabling small, accurate movements, and the buttons providing enough inputs to control all axes. Additionally, many people are at least somewhat familiar with the use of a gamepad in this style, making the robot control more intuitive than with a specialized device.

B.2 Experiment Protocol

This section contains the detailed protocols written during the execution of the various experiments.

#	Task	Target Object	Camera Position	Action Interpretation	Outcome	Amount of Movement
1.1	Pick up the X	Banana	Frontal	Absolute	failure	low
1.2	Pick up the X	Banana	Frontal	Absolute	failure	low
2.1	Pick up the X	Banana	Shoulder	Relative	failure	high
2.2	Pick up the X	Banana	Shoulder	Relative	failure	low
3.1	Pick up the X	Banana	Side	Absolute	failure	high
3.2	Pick up the X	Banana	Side	Absolute	failure	high
4.1	Pick up the X	Banana	Frontal	Relative	failure	low
4.2	Pick up the X	Banana	Frontal	Relative	failure	low
5.1	Pick up the X	Banana	Shoulder	Absolute	failure	high
5.2	Pick up the X	Banana	Shoulder	Absolute	failure	low
6.1	Pick up the X	Banana	Side	Relative	failure	high
6.2	Pick up the X	Banana	Side	Relative	failure	low
7.1	Pick up the X	Coke Can	Frontal	Absolute	failure	low
7.2	Pick up the X	Coke Can	Frontal	Absolute	failure	high
8.1	Pick up the X	Coke Can	Shoulder	Relative	failure	low
8.2	Pick up the X	Coke Can	Shoulder	Relative	failure	low
9.1	Pick up the X	Coke Can	Side	Absolute	failure	low
9.2	Pick up the X	Coke Can	Side	Absolute	failure	low
10.1	Pick up the X	Coke Can	Frontal	Relative	failure	low
10.2	Pick up the X	Coke Can	Frontal	Relative	failure	high
11.1	Pick up the X	Coke Can	Shoulder	Absolute	failure	low
11.2	Pick up the X	Coke Can	Shoulder	Absolute	failure	low
12.1	Pick up the X	Coke Can	Side	Relative	failure	high
12.2	Pick up the X	Coke Can	Side	Relative	failure	high
13.1	Place the X in the pan.	Banana	Frontal	Absolute	failure	low
13.2	Place the X in the pan.	Banana	Frontal	Absolute	failure	low
14.1	Place the X in the pan.	Banana	Shoulder	Relative	failure	low
14.2	Place the X in the pan.	Banana	Shoulder	Relative	failure	low
15.1	Place the X in the pan.	Banana	Side	Absolute	failure	high
15.2	Place the X in the pan.	Banana	Side	Absolute	failure	high
16.1	Place the X in the pan.	Banana	Frontal	Relative	failure	low
16.2	Place the X in the pan.	Banana	Frontal	Relative	failure	low
17.1	Place the X in the pan.	Banana	Shoulder	Absolute	failure	low
17.2	Place the X in the pan.	Banana	Shoulder	Absolute	failure	low
18.1	Place the X in the pan.	Banana	Side	Relative	failure	high
18.2	Place the X in the pan.	Banana	Side	Relative	failure	high
19.1	Place the X in the pan.	Coke Can	Frontal	Absolute	failure	low
19.2	Place the X in the pan.	Coke Can	Frontal	Absolute	failure	low
20.1	Place the X in the pan.	Coke Can	Shoulder	Relative	failure	low
20.2	Place the X in the pan.	Coke Can	Shoulder	Relative	failure	low
21.1	Place the X in the pan.	Coke Can	Side	Absolute	failure	high
21.2	Place the X in the pan.	Coke Can	Side	Absolute	failure	high
22.1	Place the X in the pan.	Coke Can	Frontal	Relative	failure	low
22.2	Place the X in the pan.	Coke Can	Frontal	Relative	failure	low
23.1	Place the X in the pan.	Coke Can	Shoulder	Absolute	failure	low
23.2	Place the X in the pan.	Coke Can	Shoulder	Absolute	failure	low
24.1	Place the X in the pan.	Coke Can	Side	Relative	failure	high
24.2	Place the X in the pan.	Coke Can	Side	Relative	failure	high

Table B.1: Protocol of EX1: RT-1-X zero-shot on the UMI embodiment

#	Target Position	Outcome	Terminated	Steps	Observations
1	left, middle	success	yes	23	
2	left, middle	near miss	yes	38	
3	left, middle	fail	no	50	completely off
4	left, middle	success	yes	32	first, already banana in gripper, moved away before closing. then second attemt successful. termination only after the actual successful attempt.
5	left, back	near miss	yes	39	
6	left, middle	success	yes	22	
7	left, middle	near miss	yes	22	
8	left, middle	success	yes	34	
9	left, front	success	yes	43	
10	left, front	near miss	yes	43	slightly too far back
11	left, front	near miss	yes	57	slightly too far back
12	right, middle	fail	no	50	completely stuck after a few steps
13	right, middle	near miss	no	50	moved to the middle last minute when already between grippers
14	right, middle	success	yes	50	first, grab attempt far away from target, but no terminate. second try success and then terminate
15	right, back	near miss	yes	24	
16	right, back	near miss	yes	28	
17	right, middle	fail	no	50	first seemingly random movement, then a few attempts, but never close enough
18	right, back	fail	no	50	pick up attempt around step 40, but too far away
19	right, front	near miss	no	50	
20	right, front	near miss	yes	46	two near miss attempts, terminate after second one
21	right, front	fail	no	50	just froze after step 3
22	middle, back	near miss	yes	23	off to left
23	middle, back	near miss	yes	22	off to left
24	middle, back	near miss	yes	24	off to left
25	middle, middle	near miss	yes	23	off to right
26	middle,middle	success	yes	20	
27	middle,middle	near miss	yes	23	near miss attempt, but more than 5cm away
28	middle, front	near miss	yes	25	
29	middle, front	near miss	yes	25	
30	middle, front	fail	no	50	one near miss attempt, but too far away

Table B.2: Protocol of EX2: RT-1-X-UMI evaluated in the fine-tuning scenario

#	Target Position	Outcome	Terminated	Steps	Observations
1	left, back	near miss	yes	34	very very close around step 18, then a few worse attemts
2	left, back	fail	no	50	attempt very far away
3	left, back	fail	no	50	attempt very far away
4	left, middle	fail	no	50	attempt very far away
5	left, middle	fail	yes	20	close attempt, but did not open gripper
6	left, middle	fail	no	50	a few attempts, but very far away and not opening gripper
7	left, middle	fail	no	50	seemingly random movement
8	left, front	fail	no	50	attempt far away and without opening gripper
9	left, front	near miss	yes	18	
10	left, front	fail	no	50	attempt far away
11	middle, back	near miss	yes	15	
12	middle, back	success	yes	16	
13	middle, back	fail	no	50	near miss attempt after 15 steps, but no termination
14	middle, middle	fail	no	50	somewhat near miss attempt, but a bit far away and no termination
15	middle, middle	fail	no	50	2 near miss attempts but no termination
16	middle, middle	fail	no	50	movements close to target object, but no real attempt
17	middle, middle	success	yes	15	
18	middle, front	fail	no	50	froze after step 20
19	middle, front	fail	no	50	movement close to target
20	middle, front	fail	no	50	froze after step 30
21	right, back	fail	no	50	movement close to target
22	right, back	fail	no	50	movement close to target
23	right, back	fail	no	50	movement close to target
24	right, middle	fail	no	50	somewhat near miss attempt, no termination
25	right, middle	near miss	yes	34	
26	right, middle	near miss	yes	20	
27	right, middle	success	yes	35	success on second attempt
28	right, front	near miss	yes	21	
29	right, front	near miss	yes	19	
30	right, front	fail	no	50	attempt, but a bit far away

Table B.3: Protocol of EX6: RT-1-UMI (without Open X-Embodiment pre-training) evaluated on the fine-tuning scenario

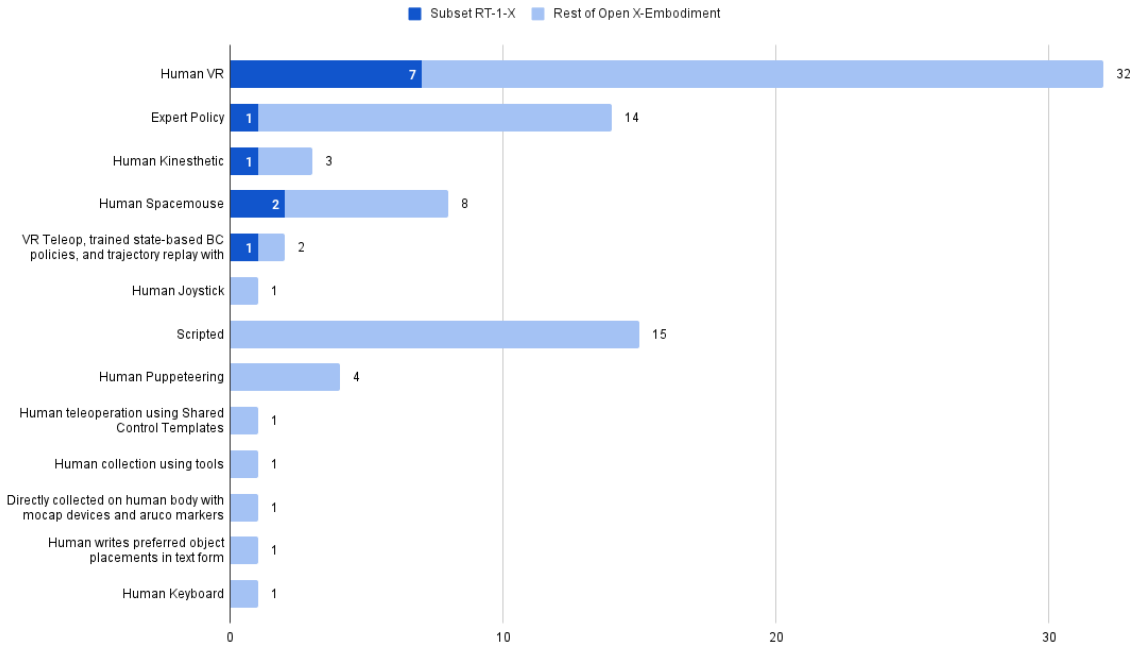

Figure B.1: The methods of data collection used in the different datasets of Open X-Embodiment. The red parts indicates the subset of Open X-Embodiment that was used in the training of RT-1-X.

#	Object	Target Position	Outcome	Termination	Steps	Observation
1	Pick up the coke can	left	near miss	yes	29	
2	Pick up the coke can	left	fail	no	50	tries to pick it up multiple times, but without termination
3	Pick up the coke can	left	fail	no	50	near miss at 28 but does not terminate
4	Pick up the coke can	middle	fail	no	50	near miss at 25 but does not terminate
5	Pick up the coke can	middle	near miss	yes	31	
6	Pick up the coke can	middle	fail	no	50	froze
7	Pick up the coke can	middle	near miss	yes	33	moved to the right position but from the side, so target object was pushed out of the way
8	Pick up the coke can	right	success	yes	25	
9	Pick up the coke can	right	near miss	yes	30	
10	Pick up the coke can	right	near miss	yes	39	

Table B.4: Protocol of EX3a: RT-1-X-UMI evaluated on a target object from Open X-Embodiment (coke can), only the target object on the workspace

#	Object	Target Position	Outcome	Termination	Steps	Observation
1	Pick up the coke can	banana left, coke right	fail	yes	29	tries banana
2	Pick up the coke can	banana left, coke right	fail	no	50	seems to oscillate between the two objects
3	Pick up the coke can	banana left, coke right	fail	no	50	
4	Pick up the coke can	banana left, coke right	fail	yes	30	tries banana
5	Pick up the coke can	banana left, coke right	fail	yes	23	tries banana
6	Pick up the coke can	banana right, coke left	near miss	yes	24	
7	Pick up the coke can	banana right, coke left	near miss	yes	29	
8	Pick up the coke can	banana right, coke left	near miss	yes	25	
9	Pick up the coke can	banana right, coke left	near miss	yes	32	
10	Pick up the coke can	banana right, coke left	near miss	yes	30	

Table B.5: Protocol of EX3b: RT-1-X-UMI evaluated on a target object from Open X-Embodiment, target object and fine-tuning object on workspace

#	Task	Target Positions	Outcome	Steps	Termination	Observation
1	Place the yellow banana in the pan.	pan left, banana right	fail	50	no	tries to pick up pan multiple times (does not get a grasp)
2	Place the yellow banana in the pan.	pan right, banana left	fail	50	no	tries to pick up pan multiple times (does not get a grasp)
3	Place the yellow banana in the pan.	pan left, banana right	fail	32	yes	picks up banana, then terminates
4	Place the yellow banana in the pan.	pan right, banana left	fail	50	no	tries to pick up pan multiple times (does not get a grasp)
5	Place the yellow banana in the pan.	pan left, banana right	fail	50	no	tries to pick up pan multiple times (does not get a grasp)
6	Place the yellow banana in the pan.	pan left, banana right	fail	50	no	tries to pick up banana
7	Place the yellow banana in the pan.	pan right, banana left	fail	50	no	tries to pick up pan multiple times (does not get a grasp)
8	Place the yellow banana in the pan.	pan left, banana right	fail	50	no	movements between the two objects
9	Place the yellow banana in the pan.	pan right, banana left	fail	50	no	tries to pick up pan multiple times (does not get a grasp)
10	Place the yellow banana in the pan.	pan right, banana left	fail	50	no	tries to pick up pan multiple times (does not get a grasp)

Table B.6: Protocol of EX4: RT-1-X-UMI evaluated on a task from Open X-Embodiment

#	Target position	Outcome	Termination	Steps	Observation
1	left		no	50	pickup attempt very far away
2	left		no	50	random movement
3	left		no	50	random movement
4	middle		yes	21	movement around the right area
5	middle		yes	24	pickup attempt very far away
6	middle		yes	24	random movement, then termination
7	midde		yes	27	pickup attempt very far away
8	right	near miss	yes	37	
9	right		yes	9	kind of near miss, but didnt open gripper
10	right		no	50	froze

Table B.7: Protocol for EX5: RT-1-X-UMI evaluated on fine-tuning task with changed environment