

Learning Efficiently with Trajectory Data for Real World Robotics

Philipp Wu

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/Eecs-2024-209

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-209.html>

December 10, 2024



Copyright © 2024, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Learning Efficiently with Trajectory Data for Real World Robotics

By

Philipp Wu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Pieter Abbeel, Chair

Professor Ken Goldberg

Professor Sergey Levine

Fall 2024

Learning Efficiently with Trajectory Data for Real World Robotics

Copyright 2024
by
Philipp Wu

Abstract

Learning Efficiently with Trajectory Data for Real World Robotics

by

Philipp Wu

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Pieter Abbeel, Chair

With the technological advancements enabled by AI, the vision of generally capable robots is now within reach. In this dissertation, I discuss my work on leveraging data-driven learning approaches for real-world robotic systems, centering on trajectory data—the complex, multi-modal, time-series information that serves as the core unit of data in robotics. The data sources in robotics are complex, potentially coming from multiple sources with varying quality. Additionally, collecting real-world robot data can be expensive and time-consuming, making efficient use of each data point essential.

I aim to address these challenges through three key research directions: trajectory representation learning, high-quality data collection, and sample-efficient policy learning. First, we explore how to learn effective representations from trajectory data, using both reconstruction-based and contrastive learning methods, and demonstrate how these representations enhance a variety of downstream robotics tasks. Next, we examine practical methods that can be used with real-world robotic systems to collect high-quality trajectory data and subsequently utilize that data to learn new skills. Finally, we investigate how to compose these robot skills with high-level language models to imbue robots with stronger reasoning and planning capabilities.

Together, these contributions advance the development of general-purpose robots capable of operating in complex, unstructured environments.

To my family.

Contents

Contents	ii
1 Introduction	1
1.1 Vision	1
1.2 Trajectory Representation Learning	2
1.3 Low-level Policy Learning on Real-World Robots	2
1.4 Coordinating Robot Policies with Language Models	2
1.5 Concluding Thoughts	3
2 Trajectory Representations with Masking	4
2.1 Introduction	5
2.2 Related Work	7
2.3 Masked Trajectory Modeling	8
2.4 Experiments	11
2.5 Summary	18
3 Trajectory Clustering for Imitation Learning	19
3.1 Introduction	19
3.2 Related work	21
3.3 Problem Formulation	23
3.4 Method	24
3.5 Experiments	26
3.6 Conclusions	30
4 Robot Learning with World Models	32
4.1 Introduction	33
4.2 Approach	35
4.3 Experiments	37
4.4 Related Work	42
4.5 Discussion	43
5 Trajectory Data Collection for Manipulation	45

5.1	Introduction	46
5.2	Related Work	47
5.3	Teleoperation Device Design	49
5.4	Experiments	53
5.5	Discussion	56
6	Robot Policy Learning with Human-in-the-Loop	58
6.1	Introduction	59
6.2	Related Work	60
6.3	Continual Interactive Imitation Learning	62
6.4	Teleoperation System	63
6.5	Experiments	65
6.6	Discussion	70
7	Task Planning with Language Models	72
7.1	Introduction	72
7.2	Related Work	74
7.3	Method	76
7.4	Experiments	79
7.5	Conclusion	83
8	Hierarchical Policies with Language Models	84
8.1	Introduction	84
8.2	Related Work	86
8.3	Method	87
8.4	Results	92
8.5	Conclusion	96
9	Conclusion	97
	Bibliography	99
A	Chapter 2 Appendix	126
A.1	Additional MTM Results	126
A.2	Additional Environment Details	128
A.3	Model and Training Details	129
A.4	Effect of training trajectory length	130
A.5	Additional plots	131
B	Chapter 3 Appendix	133
B.1	Appendix	133
C	Chapter 4 Appendix	135

C.1	Adaptation	135
C.2	Imagination	136
C.3	Detailed Related Work	137
C.4	Environment and Hardware Details	138
C.5	Hyperparameters	139
D	Chapter 6 Appendix	140
D.1	Additional Real-world Task Visualization	140
D.2	Simulation Task Visualizations	141
D.3	Robot Hardware Details	141
D.4	Teleoperation Capabilities	145
E	Chapter 8 Appendix	146
E.1	GPT-4V Prompting Details	146
E.2	Dataset Details For Language Table	147

Acknowledgments

Completing this Ph.D. has been an incredible journey. Reaching this destination has only been possible with the guidance, support, and encouragement of the many extraordinary individuals in my life.

First and foremost, I would like to express my sincere gratitude to my advisor, Pieter Abbeel. Pieter has been an exceptional mentor and an enduring source of inspiration throughout my academic career. From fostering my research capabilities as an undergraduate to welcoming me into his lab as a Ph.D. student, he has created an environment where I could thrive and deepen my research passions. The opportunity to work on impactful and exciting projects under his guidance has profoundly shaped both my academic and personal trajectory. Pieter's steady support has been invaluable to me every step of the way.

I am grateful to my dissertation and qualification exam committee members, Pieter Abbeel, Sergey Levine, Ken Goldberg, and Aravind Rajeswaran. I am honored to have the world's leading researchers in AI and robotics on my committee and it has been an absolute highlight of my academic journey. Thank you to Aravind, for hosting me at Meta during my time there as a student researcher and for providing insightful encouragement and mentorship. My heartfelt appreciation also goes to Ken for serving as the chair for my qualifying exam committee and being encouraging of my research over the years.

I extend my gratitude to all the collaborators I have had the pleasure of working with throughout the years. The collaborative nature of research has always been a source of joy and intellectual stimulation for me. Thank you, Pieter Abbeel, Ken Goldberg, Aravind Rajeswaran, Danjar Hafner, Alejandro Escontrela, Yide Shentu, Kevin Zakka, Xingyu Lin, Boyi Li, Kourosh Hakhmaneshi, Laura Smith, Qiayuan Liao, Jitendra Malik, Igor Mordatch, Yuqing Du, Zhongke Yi, Arjun Majumdar, Yixin Lin, Kevin Stone and many others. Working alongside such brilliant and talented individuals has been a rewarding and enriching experience and I've learned so much from all of you.

I would also like to thank my labmates for fostering an atmosphere of camaraderie, creativity, and mutual support. Whether tackling complex problems or sharing laughs during lab socials, these shared experiences have enriched my time in the lab and left me with unforgettable memories.

To my friends, your encouragement, patience, and unwavering belief in me have been a constant source of strength. You've made the challenges of deadlines, conferences, and late nights manageable, and the milestones all the more meaningful. Thank you for sticking it through the tough times and sharing the happy moments with me.

Finally, to my family, I owe my deepest gratitude. To my mom and dad, thank you for your boundless love, for encouraging me to pursue my dreams, and for instilling in me the confidence to achieve them. To my sister, Elise, thank you for being a constant pillar of support I can always rely on. Your unwavering encouragement has meant everything to me.

To everyone who has contributed to this journey, thank you for shaping not only this dissertation but also my growth as a researcher and as a person. I am excited to explore the future with gratitude for all that has brought me here.

Chapter 1

Introduction

1.1 Vision

My research goal is to create generally intelligent robotic systems capable of operating in unstructured environments, spanning the low-level control systems to the high-level intelligence and learning algorithms. I feel incredibly fortunate to have pursued my Ph.D. during such an exciting period of technological advancement in AI and robotics. As a child, I spent hours building robots with my LEGO robotics kit, programming them to perform simple autonomous behaviors. Now, through my Ph.D., I have had the privilege of contributing to the scientific community and advancing robotics toward a new frontier of embodied agents with general capabilities powered by deep learning.

The core of my work centers on trajectory data, the special form of data that we manipulate for robotics. A trajectory is a form of time-series data, encompassing multiple modalities. For instance, a humanoid robot might have data in the form of proprioception from joint encoders, visual observations from cameras, pressure readings from foot sensors, and audio signals from a microphone. While this diversity and multimodality make trajectory data both rich and valuable for learning, it also introduces significant complexity. Furthermore, the hardware requirements and physical constraints of robotics present additional challenges, particularly in collecting sufficient high-quality data for data-hungry algorithms.

This dissertation explores methods to address these challenges, investigating ways to learn from trajectory data in the real world and ultimately contributing to the development of general-purpose robots. We explore three main directions:

1. Develop algorithms that extract the maximum value from each data point through representation learning.
2. Design methods to gather high-quality trajectory data in the real world and learn robot policies from that data.
3. Leverage the high-level reasoning priors of language models to coordinate with low-level control policies.

1.2 Trajectory Representation Learning

By effectively pretraining representations on trajectory data, we can greatly benefit downstream robot learning. However, trajectory data are very complex, coming from a variety of different sources with different modalities. In Chapter 2, we explore Masked Trajectory Models, which address this challenge by learning representations on trajectory data using a masked training and reconstruction approach. We find this benefits policy learning in a variety of ways. In Chapter 3, we further investigate how representations help in policy learning through trajectory clustering, which we apply to a semi-supervised one-shot imitation learning setting, this time through a contrastive-based approach. This section provides tools for learning representations from existing trajectory data.

1.3 Low-level Policy Learning on Real-World Robots

This section investigates how to actually collect trajectory datasets in the real world, and develop algorithms to learn effective policies from these datasets. Real-world data collection for robotics presents unique challenges. One approach involves enabling robots to autonomously collect data through trial and error, learning about their environment and tasks. We explore this in Chapter 4 in the work *Daydreamer*, where we propose using world models—learned latent dynamics models that allow agents to predict the outcomes of their actions—to improve data efficiency. We ultimately learn policies across 4 different robots directly with trajectory data collected from the robots’ own experience. Despite these advancements, challenges remain, particularly in robot manipulation tasks, due to the difficulty of acquiring rewards in the real world. In imitation learning, robots can benefit from expert demonstrations, circumventing the need for rewards. In Chapter 5, we introduce GELLO, a low-cost and accessible teleoperation solution, which facilitates high-quality data collection for imitation learning. This system enables intuitive teleoperation, making demonstration dataset collection scalable and accessible. Chapter 6 explores further improving this process of imitation learning by having humans provide corrective feedback during policy execution. Our practical robot system, *RoboCopilot*, reduces human effort while improving data efficiency by specifically targeting failure modes. This section focuses on collecting trajectory data and training useful real world policies from such data

1.4 Coordinating Robot Policies with Language Models

This section explores the high-level capabilities of language models and how they can benefit robot systems by providing an interface for humans to interact with. Large language models (LLMs) excel at high-level reasoning and task planning, but often struggle with low-level tasks that are represented in trajectory data, requiring fine motor control and precision.

This limitation necessitates integration with specialized control systems to achieve a balance between abstraction and practical execution. Chapter 7 introduces a framework where LLMs serve as interpreters and planners, generating actionable task plans and coordinating subtasks with low-level skills. This system facilitates iterative dialogue, ensuring alignment with user expectations and situational requirements. Chapter 8 further explores this concept and presents a hierarchical framework that combines the strengths of high-level reasoning through vision-language models with the precision of low-level control policies through end-to-end training. This approach achieves seamless coordination between abstract planning and physical execution. This section advances the potential for human-robot collaboration through language interfaces, paving the way for more adaptive and interactive robotic systems that can reason.

1.5 Concluding Thoughts

We conclude this dissertation in Chapter 9 with some closing thoughts on the various components we explored around trajectory data. We then touch upon the open-ended problems we need to address to truly develop general robotic systems that can function in unstructured environments like humans and present an outlook for the future of robotics.

Chapter 2

Trajectory Representations with Masking

This chapter is based on the paper “Masked Trajectory Models for Prediction, Representation, and Control” (Wu et al., 2023d), by Philipp Wu, Arjun Majumdar, Kevin Stone, Yixin Lin, Igor Mordatch, Pieter Abbeel, and Aravind Rajeswaran.

We introduce Masked Trajectory Models (MTM) as a generic abstraction for sequential decision making. MTM takes a trajectory, such as a state-action sequence, and aims to reconstruct the trajectory conditioned on random subsets of the same trajectory. By training with a highly randomized masking pattern, MTM learns versatile networks that can take on different roles or capabilities, by simply choosing appropriate masks at inference time. For

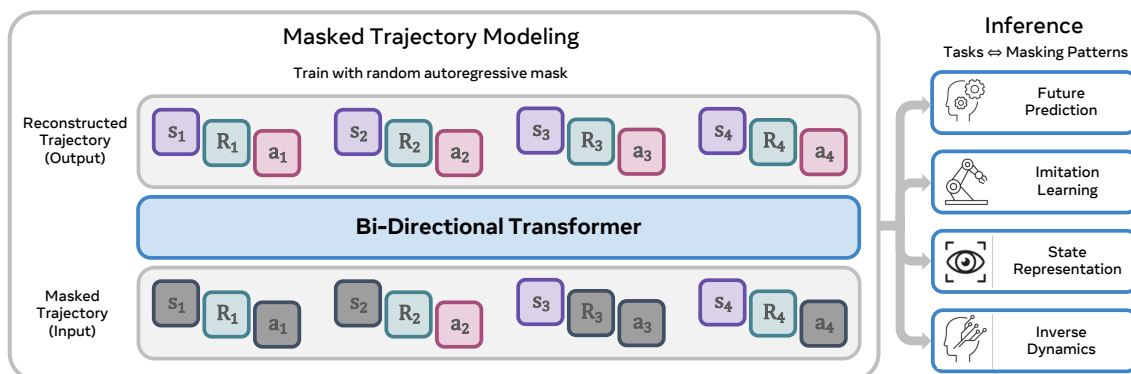


Figure 2.1: **Masked Trajectory Modeling (MTM) Framework.** (Left) The training process involves reconstructing trajectory segments from a randomly masked view of the same. (Right) After training, MTM can enable several downstream use-cases by simply changing the masking pattern at inference time. See Section 2.3 for discussion on training and inference masking patterns.

example, the same MTM network can be used as a forward dynamics model, inverse dynamics model, or even an offline RL agent. Through extensive experiments in several continuous control tasks, we show that the same MTM network – i.e. same weights – can match or outperform specialized networks trained for the aforementioned capabilities. Additionally, we find that state representations learned by MTM can significantly accelerate the learning speed of traditional RL algorithms. Finally, in offline RL benchmarks, we find that MTM is competitive with specialized offline RL algorithms, despite MTM being a generic self-supervised learning method without any explicit RL components. Code is available at <https://github.com/facebookresearch/mtm>.

2.1 Introduction

Sequential decision making is a field with a long and illustrious history, spanning various disciplines such as reinforcement learning (Sutton & Barto, 2018), control theory (Bertsekas, 1995; Åström & Murray, 2010), and operations research (Powell, 2007). Throughout this history, several paradigms have emerged for training agents that can achieve long-term success in unknown environments. However, many of these paradigms necessitate the learning and integration of multiple component pieces to obtain decision-making policies. For example, model-based RL methods require the learning of world models and actor-critic methods require the learning of critics. This leads to complex and unstable multi-loop training procedures and often requires various ad-hoc stabilization techniques. In parallel, the emergence of self-supervised learning (Devlin et al., 2018; Jing & Tian, 2019) has led to the development of simple training objectives such as masked prediction and contrastive prediction, which can train generic backbone models for various tasks in computer vision and natural language processing (NLP). Motivated by this advancement, we explore if self-supervised learning can lead to the creation of generic and versatile models for sequential decision making with capabilities including future prediction, imitation learning, and representation learning.

Towards this end, we propose the use of Masked Trajectory Models (MTM) as a generic abstraction and framework for prediction, representation, and control. Our approach draws inspiration from two recent trends in Artificial Intelligence. The first is the success of masked prediction, also known as masked autoencoding, as a simple yet effective self-supervised learning objective in NLP (Devlin et al., 2018; Liu et al., 2019; Brown et al., 2020) and computer vision (Bao et al., 2021; He et al., 2021). This task of masked prediction not only forces the model to learn good representations but also develops its conditional generative modeling capabilities. The second trend that inspires our work is the recent success of transformer sequence models, such as decision transformers, for reinforcement (Chen et al., 2021; Janner et al., 2021) and imitation learning (Reed et al., 2022; Shafiullah et al., 2022). Motivated by these breakthroughs, we investigate if the combination of masked prediction and transformer sequence models can serve as a generic self-supervised learning paradigm for decision-making.

Conceptually, MTM is trained to take a trajectory sequence of the form:

$$\boldsymbol{\tau} := (\mathbf{s}_k, \mathbf{a}_k, \mathbf{s}_{k+1}, \mathbf{a}_{k+1}, \dots, \mathbf{s}_t, \mathbf{a}_t)$$

and reconstruct it given a masked view of the same, i.e.

$$\hat{\boldsymbol{\tau}} = \mathbf{h}_\theta(\text{Masked}(\boldsymbol{\tau}))$$

where $\mathbf{h}_\theta(\cdot)$ is a bi-directional transformer and $\text{Masked}(\boldsymbol{\tau})$ is a masked view of $\boldsymbol{\tau}$ generated by masking or dropping some elements in the sequence. For example, one masked view of the above sequence could be: $(\mathbf{s}_k, _, _, \mathbf{a}_{k+1}, _, \dots, \mathbf{s}_t, _)$ where $_$ denotes a masked element. In this case, MTM must infill intermediate states and actions in the trajectory as well as predict the next action in the sequence. A visual illustration of our paradigm is shown in Figure 2.1. Once trained, MTM can take on multiple roles or capabilities at inference time by appropriate choice of masking patterns. For instance, by unmasking actions and masking states in the sequence, MTM can function as a forward dynamics model.

Our Contributions Our main contribution is the proposal of MTM as a versatile modeling paradigm and pre-training method. We empirically investigate the capabilities of MTM on several continuous control tasks including planar locomotion (Fu et al., 2020) and dexterous hand manipulation (Rajeswaran et al., 2018). We highlight key findings and unique capabilities of MTM below.

1. **One Model, Many Capabilities:** The same model trained with MTM (i.e. the same set of weights) can be used zero-shot for multiple purposes including inverse dynamics, forward dynamics, imitation learning, offline RL, and representation learning.
2. **Heteromodality:** MTM is uniquely capable of consuming heteromodal data and performing missing data imputation, since it was trained to reconstruct full trajectories conditioned on randomly masked views. This capability is particularly useful when different trajectories in the dataset contain different modalities, such as a dataset containing both state-only trajectories as well as state-action trajectories (Baker et al., 2022). Following the human heteromodal cortex (Donnelly, 2011), we refer to this capability as heteromodality.
3. **Data Efficiency:** Training with random masks enables different training objectives or combinations, thus allowing more learning signal to be extracted from any given trajectory. As a result, we find MTM to be more data efficient compared to other methods.
4. **Representation Learning:** We find that state representations learned by MTM transfer remarkably well to traditional RL algorithms like TD3 (Fujimoto et al., 2018b), allowing them to quickly reach optimal performance. This suggests that MTM can serve as a powerful self-supervised pre-training paradigm, even for practitioners who prefer to use conventional RL algorithms.

Overall, these results highlight the potential for MTM as a versatile paradigm for RL, and its ability to be used as a tool for improving the performance of traditional RL methods.

2.2 Related Work

Autoencoders and Masked Prediction. Autoencoders have found several applications in machine learning. The classical PCA (Jolliffe & Cadima, 2016) can be viewed as a linear autoencoder. Denoising autoencoders (Vincent et al., 2008) learn to reconstruct inputs from noise corrupted versions of the same. Masked autoencoding has found recent success in domains like NLP (Devlin et al., 2018; Brown et al., 2020) and computer vision (He et al., 2021; Bao et al., 2021). Our work explores the use of masked prediction as a self-supervised learning paradigm for RL.

Offline Learning for Control Our work primarily studies the offline setting for decision making, where policies are learned from static datasets. This broadly falls under the paradigm of offline RL (Lange et al., 2012). A large class of offline RL algorithms modify their online counterparts by incorporating regularization to guard against distribution shift that stems from the mismatch between offline training and online evaluation (Kumar et al., 2020; Kidambi et al., 2020; Fujimoto et al., 2018a; Yu et al., 2021; Liu et al., 2020). In contrast, our work proposes a generic self-supervised pre-training paradigm for decision making, where the resulting model can be directly repurposed for offline RL. Zheng et al. (2023) introduces a self supervised approach for the heteromodal offline RL settings where only a small subset of the trajectories have action labels. We leverage this setting in the investigation of Heteromodal MTM, which can be trained without any change to the algorithm.

Self-Supervised Learning for Control The broad idea of self-supervision has been incorporated into RL in two ways. The first is self-supervised **data collection**, such as task-agnostic and reward-free exploration (Pathak et al., 2017; Laskin et al., 2021; Burda et al., 2018). The second is concerned with self-supervised **learning** for control, which is closer to our work. Prior works typically employ self-supervised learning to obtain state representations (Yang & Nachum, 2021; Parisi et al., 2022; Nair et al., 2022; Xiao et al., 2022) or world models (Hafner et al., 2020; Hansen et al., 2022a,b; Seo et al., 2022), for subsequent use in standard RL pipelines. In contrast, MTM uses self-supervised learning to train a single versatile model that can exhibit multiple capabilities.

Transformers and Attention in RL Our work is inspired by the recent advances in AI enabled by transformers (Vaswani et al., 2017), especially in offline RL (Chen et al., 2021; Janner et al., 2021; Jiang et al., 2022) and imitation learning (Reed et al., 2022; Shafiq et al., 2022; Brohan et al., 2022; Jiang et al., 2023; Zhou et al., 2022). Of particular relevance are works that utilize transformers in innovative ways beyond the standard RL paradigm. Decision Transformers and related methods (Schmidhuber, 2019; Srivastava et al., 2019; Chen et al., 2021) use return-conditioned imitation learning, which we also adopt in this work. However, in contrast to Chen et al. (2021) and Janner et al. (2021) who use next token prediction as the self-supervised task, we use a bi-directional masked prediction objective.

This masking pattern enables the learning of versatile models that can take on different roles based on inference-time masking pattern.

Recently, Liu et al. (2023a) and Carroll et al. (2022) explore the use of bi-directional transformers for RL and we build off their work. In contrast to Liu et al. (2023a) which studies downstream tasks like goal reaching and skill prompting, we study a different subset of tasks such as forward and inverse dynamics. Liu et al. (2023a) also studies offline RL by applying TD3 and modifying the transformer attention mask to be causal, while we study the return conditioned behavior cloning setting. In contrast to Carroll et al. (2022), we study the broader capabilities of our model on several high-dimensional control tasks. VPT (Baker et al., 2022) also tackles sequential decision making using transformers, focusing primarily on extracting action labels with a separate inverse dynamics model. Furthermore, unlike prior work, we also demonstrate that our model has unique and favorable properties like data efficiency, heteromodality, and the capability to learn good state representations.

2.3 Masked Trajectory Modeling

We now describe the details of our masked trajectory modeling paradigm, such as the problem formulation, training objective, masking patterns, and overall architecture used.

2.3.1 Trajectory Datasets

MTM is designed to operate on trajectory datasets that we encounter in decision making domains. Taking the example of robotics, a trajectory comprises of proprioceptive states, camera observations, control actions, task/goal commands, and so on. We can denote such a trajectory comprising of M different modalities as

$$\tau = \left\{ \left(\mathbf{x}_1^1, \mathbf{x}_1^2, \dots, \mathbf{x}_1^M \right), \dots, \left(\mathbf{x}_T^1, \mathbf{x}_T^2, \dots, \mathbf{x}_T^M \right) \right\}, \quad (2.1)$$

where \mathbf{x}_t^m refers to the m^{th} modality in the t^{th} timestep. In our empirical investigations, following prior work (Chen et al., 2021; Janner et al., 2021), we use state, action, and return-to-go (RTG) sequences as the different data modalities. Note that in-principle, our mathematical formulation is generic and can handle any modality.

2.3.2 Architecture and Masked Modeling

To perform masked trajectory modeling, we first “tokenize” the different elements in the raw trajectory sequence, by lifting them to a common representation space using modality-specific encoders. Formally, we compute

$$\mathbf{z}_t^m = E_\theta^m(\mathbf{x}_t^m) \quad \forall t \in [1, T], m \in [1, M],$$

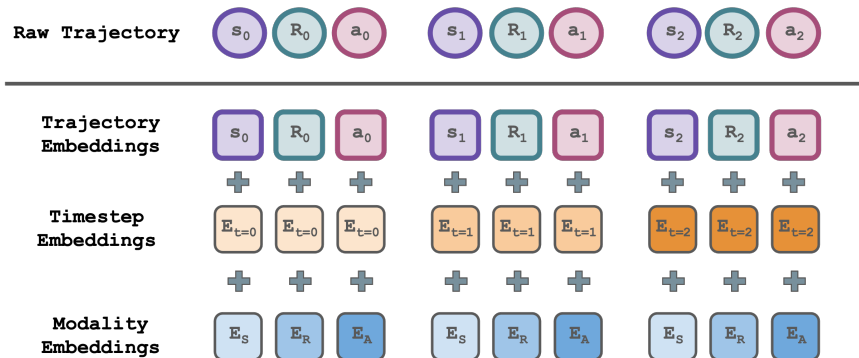


Figure 2.2: **Tokenization of the trajectory sequence** comprises three components. A modality specific encoder lifts from the raw modality space to a common representation space, where we additionally add timestep embeddings and modality type embeddings. Collectively, these allow the transformer to distinguish between different elements in the sequence.

where E_θ^m is the encoder corresponding to modality m . We subsequently arrange the embeddings in a 1-D sequence of length $N = M \times T$ as:

$$\boldsymbol{\tau} = (\mathbf{z}_1^1, \mathbf{z}_1^2, \dots, \mathbf{z}_1^M, \dots, \mathbf{z}_t^m, \dots, \mathbf{z}_T^M).$$

The self-supervised learning task in MTM is to reconstruct the above sequence conditioned on a masked view of the same. We denote the latter with $\text{Masked}(\boldsymbol{\tau})$, where we randomly drop or “mask” a subset of elements in the sequence. The final self-supervised objective is given by:

$$\max_{\theta} \mathbb{E}_{\boldsymbol{\tau}} \sum_{t=1}^T \sum_{m=1}^M \log P_{\theta}(\mathbf{z}_t^m | \text{Masked}(\boldsymbol{\tau})), \quad (2.2)$$

where P_{θ} is the prediction of the model. This encourages the learning of a model that can reconstruct trajectories from parts of it, forcing it to learn about the environment as well as the data generating policy, in addition to good representations of the various modalities present in the trajectory.

Architecture and Embeddings We adopt an encoder-decoder architecture similar to He et al. (2021) and Liu et al. (2023a), where both the encoder and decoder are bi-directional transformers. We use a modality-specific encoder to lift the raw trajectory inputs to a common representation space for tokens. Further, to allow the transformer to disambiguate between different elements in the sequence, a fixed sinusoidal timestep encoding and a learnable mode-specific encoding are added, as illustrated in Figure 2.2. The resulting sequence is then flattened and fed into the transformer encoder where only unmasked tokens are processed. The decoder processes the full trajectory sequence, and uses values from the encoder when available, or a mode-specific mask token when not. The decoder is trained to predict the original sequence, including the unmasked tokens, using an MSE loss (He et al.,

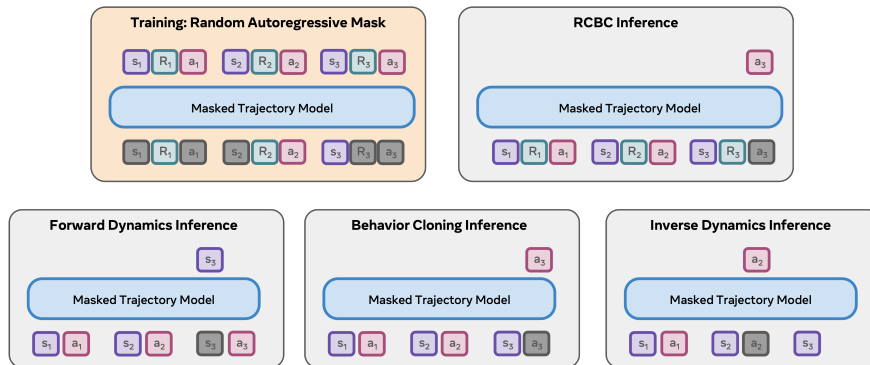


Figure 2.3: **Masking Pattern for Training and Inference.** (Training: box in orange) MTM is trained to reconstruct trajectory segments conditioned on a masked view of the same. We use a random autoregressive masking pattern, where elements in the input sequence are randomly masked, with the added constraint that at least one masked token must have no future unmasked tokens. This means the last element in the sequence must necessarily be masked. We note that the input sequence can start and end on arbitrary modalities. In this illustrated example, R_3 is the masked token that satisfies the autoregressive constraint. That is the prediction of R_3 is conditioned on no future tokens in the sequence. (Inference: boxes in gray) By changing the masking pattern at inference time, MTM can either be used directly for offline RL using RCBC (Chen et al., 2021), or be used as a component in traditional RL pipelines as a state representation, dynamics model, policy initialization, and more. These different capabilities are shown in gray. Modes not shown at the input are masked out and modes not shown at the output are not directly relevant for the task of interest.

2021), which corresponds to a Gaussian probabilistic model. We also note that the length of episodes/trajectories in RL can be arbitrarily long. In our practical implementation, we model shorter “trajectory segments” that are randomly sub-selected contiguous segments of fixed length from the full trajectory.

Masking Pattern Intuitively, we can randomly mask elements in the sequence with a sufficiently high mask ratio to make the self-supervised task difficult. This has found success in computer vision (He et al., 2021). We propose to use a variation of this – a random autoregressive masking pattern. This pattern requires at least one token in the masked sequence to be autoregressive, meaning it must be predicted based only on previous tokens, and all future tokens are masked. This means the last element in each sampled trajectory segment is necessarily masked. See Figure 2.3 for an illustration. We note that the autoregressive mask in our context is **not** using a causal mask in attention weights, but instead corresponds to masking at the input and output token level, similar to MAE.

In the case of computer vision and NLP, the entire image or sentence is often available at inference time. However, in the case of RL, the sequence data is generated as the agent

interacts with the environment. As a result, at inference time, the model is forced to be causal (i.e. use only the past tokens). By using our random autoregressive masking pattern, the model both learns the underlying temporal dependencies in the data, as well as the ability to perform inference on past events. We find that this simple modification is helpful in most tasks we study.

2.3.3 MTM as a generic abstraction for RL

The primary benefit of MTM is its versatility. Once trained, the MTM network can take on different roles, by simply using different masking patterns at inference time. We outline a few examples below. See Figure 2.3 for a visual illustration.

1. Firstly, MTM can be used as a stand-alone algorithm for offline RL, by utilizing a return-conditioned behavior cloning (RCBC) mask at inference time, analogous to DT (Chen et al., 2021) and RvS (Emmons et al., 2021). However, in contrast to DT and RvS, we use a different self-supervised pre-training task and model architecture. We find in Section 2.4.3 that using MTM in “RCBC-mode” outperforms DT and RvS.
2. Alternatively, MTM can be used to recover various components that routinely feature in traditional RL pipelines, as illustrated in Figure 2.3. Conceptually, by appropriate choice of masking patterns, MTM can: (a) provide state representation that accelerates the learning of traditional RL algorithms; (b) perform policy initialization through behavior cloning; (c) act as a world model for model-based RL algorithms; (d) act as an inverse dynamics model to recover action sequences that track desired reference state trajectories.

2.4 Experiments

Through detailed empirical evaluations, we aim to study the following questions.

1. Is MTM an effective algorithm for offline RL?
2. Is MTM a versatile learner? Can the same network trained with MTM be used for different capabilities without additional training?
3. Is MTM an effective heteromodal learner? Can it consume heteromodal datasets, like state-only and state-action trajectories, and effectively use such a dataset to improve performance?
4. Can MTM learn good representations that accelerate downstream learning with standard RL algorithms?

See Appendix for additional details about model architecture and hyperparameters.

2.4.1 Benchmark Datasets

To help answer the aforementioned questions, we draw upon a variety of continuous control tasks and datasets that leverage the MuJoCo simulator Todorov et al. (2012). Additional environment details can be found in Appendix A.2.

D4RL (Fu et al., 2020) is a popular offline RL benchmark consisting of several environments and datasets. Following a number of prior work, we focus on the locomotion subset: **Walker2D**, **Hopper**, and **HalfCheetah**. For each environment, we consider 4 different dataset settings: **Expert**, **Medium-Expert**, **Medium**, and **Medium-Replay**. The **Expert** dataset is useful for benchmarking imitation learning with BC, while the other datasets enable studying offline RL and other capabilities of MTM such as future prediction and inverse dynamics.

Adroit (Rajeswaran et al., 2018) is a collection of dexterous manipulation tasks with a simulated five-fingered. We experiment with the **Pen**, and **Door** tasks that test an agent’s ability to carefully coordinate a large action-space to accomplish complex robot manipulation tasks. We collect **Medium-Replay** and **Expert** trajectories for each task using a protocol similar to D4RL.

ExORL (Yarats et al., 2022) dataset consists of trajectories collected using various unsupervised exploration algorithms. Yarats et al. (2022) showed that TD3 (Fujimoto et al., 2018b) can be effectively used to learn in this benchmark. We use data collected by a ProtoRL agent (Yarats et al., 2021b) in the **Walker2D** environment to learn three different tasks: **Stand**, **Walk**, and **Run**.

2.4.2 Offline RL results

We first test the capability of MTM to learn policies in the standard offline RL setting. To do so, we train MTM with the random autoregressive masking pattern as described in Section 2.3. Subsequently, we use the Return Conditioned Behavior Cloning (RCBC) mask at inference Table 2.1: **Results on D4RL**. Offline RL results on the V2 locomotion suite of D4RL are reported here, specified by the normalized score as described in Fu et al. (2020). We find that **MTM** outperforms RvS and DT, which also use RCBC for offline RL.

Environment	Dataset	BC	CQL	IQL	TT	MOPO	RsV	DT	MTM (Ours)
HalfCheetah	Medium-Replay	36.6	45.5	44.2	41.9	42.3	38.0	36.6	43.0
Hopper	Medium-Replay	18.1	95.0	94.7	91.5	28.0	73.5	82.7	92.9
Walker2d	Medium-Replay	26.0	77.2	73.9	82.6	17.8	60.6	66.6	77.3
HalfCheetah	Medium	42.6	44.0	47.4	46.9	53.1	41.6	42.0	43.6
Hopper	Medium	52.9	58.5	66.3	61.1	67.5	60.2	67.6	64.1
Walker2d	Medium	75.3	72.5	78.3	79.0	39.0	71.7	74.0	70.4
HalfCheetah	Medium-Expert	55.2	91.6	86.7	95.0	63.7	92.2	86.8	94.7
Hopper	Medium-Expert	52.5	105.4	91.5	110.0	23.7	101.7	107.6	112.4
Walker2d	Medium-Expert	107.5	108.8	109.6	101.9	44.6	106.0	108.1	110.2
Average		51.9	77.6	77.0	78.9	42.2	71.7	74.7	78.7

Table 2.2: **Evaluation of various MTM capabilities.** MTM refers to the model trained with the random autoregressive mask, and evaluated using the appropriate mask at inference time. S-MTM (“Specialized”) refers to the model that uses the appropriate mask both during training and inference time. We also compare with a specialized MLP baseline trained separately for each capability. Note that higher is better for BC and RCBC, while lower is better for FD and ID. We find that MTM is often comparable or better than training on specialized masking patterns, or training specialized MLPs. We use a box outline to indicate that a single model was used for all the evaluations within it. The right most column indicates if MTM is comparable or better than S-MTM, and we find this to be true in most cases.

Domain	Dataset	Task	MLP	S-MTM (Ours)	MTM (Ours)	(MTM) \gtrsim (S-MTM)?
D4RL Hopper	Expert	(\uparrow) BC	111.14 \pm 0.33	111.81 \pm 0.18	107.35 \pm 7.77	\checkmark
	Expert	(\uparrow) RCBC	111.17 \pm 0.56	112.64 \pm 0.47	112.49 \pm 0.37	\checkmark
	Expert	(\downarrow) ID	0.009 \pm 0.000	0.013 \pm 0.000	0.050 \pm 0.026	\times
	Expert	(\downarrow) FD	0.072 \pm 0.000	0.517 \pm 0.025	0.088 \pm 0.049	\checkmark
D4RL Hopper	Medium Replay	(\uparrow) BC	35.63 \pm 6.27	36.17 \pm 4.09	29.46 \pm 6.74	\times
	Medium Replay	(\uparrow) RCBC	88.61 \pm 1.68	93.30 \pm 0.33	92.95 \pm 1.51	\checkmark
	Medium Replay	(\downarrow) ID	0.240 \pm 0.028	0.219 \pm 0.008	0.534 \pm 0.009	\times
	Medium Replay	(\downarrow) FD	2.179 \pm 0.052	3.310 \pm 0.425	0.493 \pm 0.030	\checkmark
Adroit Pen	Expert	(\uparrow) BC	62.75 \pm 1.43	66.28 \pm 3.28	61.25 \pm 5.06	\checkmark
	Expert	(\uparrow) RCBC	68.41 \pm 2.27	66.29 \pm 1.39	64.81 \pm 1.70	\checkmark
	Expert	(\downarrow) ID	0.128 \pm 0.001	0.155 \pm 0.001	0.331 \pm 0.049	\times
	Expert	(\downarrow) FD	0.048 \pm 0.002	0.360 \pm 0.020	0.321 \pm 0.048	\checkmark
Adroit Pen	Medium Replay	(\uparrow) BC	33.73 \pm 1.00	54.84 \pm 5.08	47.10 \pm 7.13	\times
	Medium Replay	(\uparrow) RCBC	41.26 \pm 4.99	57.50 \pm 3.76	58.76 \pm 5.63	\checkmark
	Medium Replay	(\downarrow) ID	0.308 \pm 0.004	0.238 \pm 0.004	0.410 \pm 0.064	\times
	Medium Replay	(\downarrow) FD	0.657 \pm 0.023	0.915 \pm 0.007	0.925 \pm 0.026	\checkmark

time for evaluation. This is inspired by DT (Chen et al., 2021) which uses a similar RCBC approach, but with a GPT model.

Our empirical results are presented in Table 2.1. We find that MTM outperforms the closest algorithms of DT and RvS, suggesting that masked prediction is an effective pre-training task for offline RL when using RCBC inference mask. More surprisingly, MTM is competitive with highly specialized and state-of-the-art offline RL algorithms like CQL (Kumar et al., 2020) and IQL (Kostrikov et al., 2021) despite training with a purely self-supervised learning objective without any explicit RL components.

2.4.3 MTM Capabilities

We next study if MTM is a versatile learner by evaluating it across four different capabilities on Adroit and D4RL datasets. We emphasize that we test these capabilities for a single MTM-model (i.e. same weights) by simply altering the masking pattern during inference time. See Figure 2.3 for a visual illustration of the inference-time masking patterns.

1. **Behavior Cloning (BC):** Predict next action given state-action history. This is a

standard approach to imitation learning as well as a popular initialization method for subsequent RL (Rajeswaran et al., 2018).

2. **Return Conditioned Behavior Cloning (RCBC)** is similar to BC, but additionally conditions on the desired Return-to-Go. Recent works (Chen et al., 2021; Emmons et al., 2021) have shown that RCBC can lead to successful policies in offline RL.
3. **Inverse Dynamics (ID)**, where we predict the action using the current and future desired state. This can be viewed as a 1-step goal-reaching policy. It has also found application in observation-only imitation learning (Radosavovic et al., 2021; Baker et al., 2022).
4. **Forward Dynamics (FD)**, where we predict the next state given history and current action. Forward dynamics models are an integral component of several model-based RL algorithms (Janner et al., 2019; Rajeswaran et al., 2020; Hafner et al., 2020).

We consider two variations of MTM. The first variant, S-MTM, trains a specialized model for each capability using the corresponding masking pattern at *train time*. The second variant, denoted simply as MTM, trains a single model using the random autoregressive mask specified in Section 2.3. Subsequently, the same model (i.e. same set of weights) is evaluated for all the four capabilities. We also compare our results with specialized MLP models for each capability. We evaluate the best checkpoint across all models and report mean and standard deviation across 4 seeds, taking the average of 20 trajectory executions per seed. For all experiments we train on 95% of the dataset and reserve 5% of the data for evaluation. For BC and RCBC results, we report the normalized score obtained during evaluation rollouts. For ID and FD, we report normalized loss values on the aforementioned 5% held-out data.

A snapshot of our results are presented in Table 2.2 for a subset of environments. Please see Appendix A.1 for detailed results on all the environments. The last column of the table indicates the performance difference between the versatile MTM and the specialized S-MTM. We find that MTM is comparable or even better than specialized masks, and also matches the performance of specialized MLP models. We suspect that specialized masks may require additional tuning of parameters to prevent overfitting or underfitting, whereas random autoregressive masking is more robust across tasks and hyperparameters.

2.4.4 Impact of Masking Patterns

We study if the masking pattern influences the capabilities of the learned model. Figure 2.4 shows that random autoregressive masking matches or outperforms purely random masking on RCBC for a spread of environments for offline RL. We note that pure random masking, as done in MAE and BERT, which focuses on only learning good representations, can lead to diminished performance for downstream capabilities. Random autoregressive masking mitigates these issues by allowing the learning of a single versatile model while still matching or even exceeding the performance of specialized masks, as seen in Table 2.2.

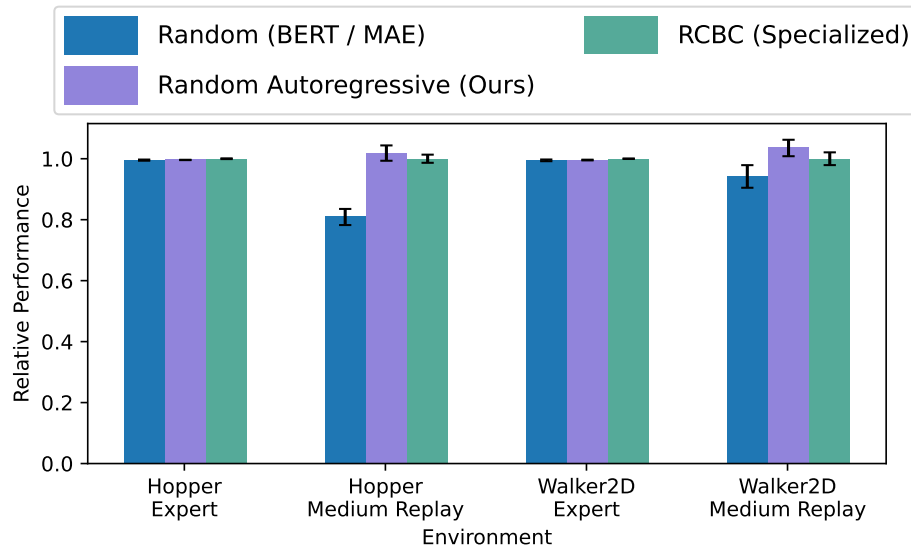


Figure 2.4: **Impact of Masking Patterns.** This plot shows MTM RCBC performance trained with three different masking patterns, random, random autoregressive, and a specialized RCBC mask. We find that autoregressive random often outperforms random, and in most cases is even competitive with the specialized (or oracle) RCBC mask. Y-axis normalized with using RCBC mask.

2.4.5 Heteromodal Datasets

MTM is uniquely capable of learning from heteromodal datasets. This is enabled by the training procedure, where any missing data can be treated as if it were masked. During training we apply the loss only to modes that exist in the dataset. For these experiments we take the **Expert** subset of our trajectory data and remove action labels from the majority of the dataset. The training data consists of 1% of the data with all modes (states, actions, return-to-go) and 95% percent of the data with no action labels. As is done in all experiments, the remainder is reserved for testing.

From our initial experiments, we found that naively adding in the state only data during training, and evaluating with the RCBC mask did not always result in improved performance. This was despite improvement in forward dynamics prediction as a result of adding state-only trajectories. Based on this observation, we propose a two-stage action inference procedure. First, we predict future states given current state and desired returns. This can be thought of as a forward dynamics pass where the desired returns are used instead of actions, which are masked out (or more precisely, missing). Next, we predict actions using the current state and predicted future states using the inverse dynamics mask. We refer to this model trained on heteromodal data, along with the two stage inference procedure, as Heteromodal MTM. We present the results in Figure 2.5, where we find that Heteromodal MTM consistently improves performance over the baseline MLP and MTM that are trained only on the subset of data with action labels.

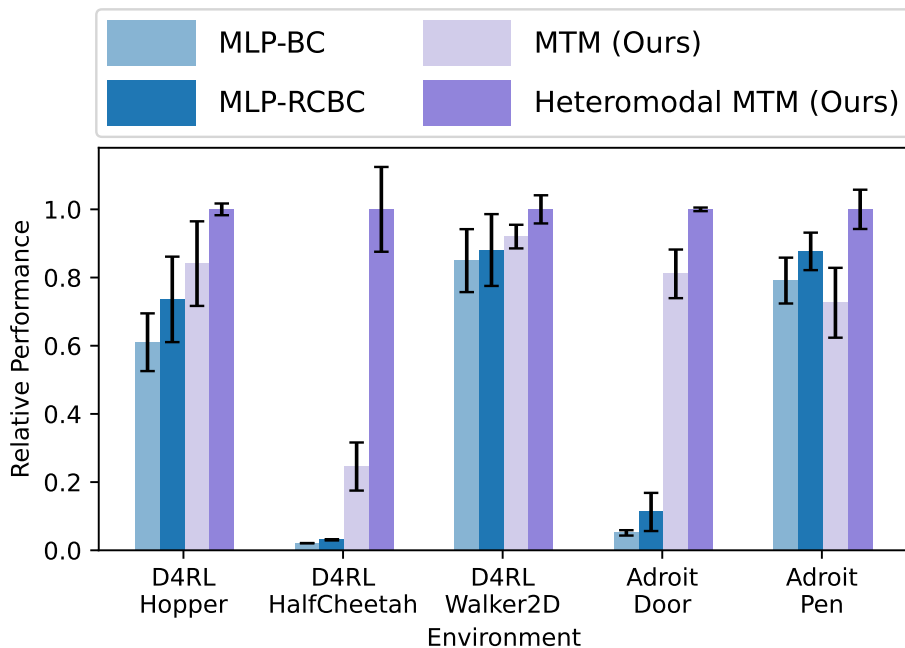


Figure 2.5: **MTM can effectively learn from heteromodal datasets.** Real world data may not always contain action labels. We simulate this setting by training a MTM models on Expert datasets across domains where only a small fraction of the data have action labels. Our Heteromodal MTM model is able to effectively improve task with the additional data over baseline MTM and MLP that train on only the subset of data with actions. Y-axis normalized with respect to performance of Heteromodal MTM.

2.4.6 Data Efficiency

Figure 2.5 not only showed the effectiveness of MTM on heteromodal data, but also that MTM is able to achieve higher performance than baseline (specialized) MLPs in the low data regimes. To explicitly test the data efficiency of MTM, we study the performance as a function of the training dataset size, and present results in Figure 2.6.

We observe that MTM is more sample efficient and achieves higher performance for any given dataset size. Heteromodal MTM also outperforms MTM throughout, with the performance gap being quite substantial in the low-data regime. We hypothesize that the data efficiency of MTM is due to better usage of the data. Specifically, since the model encounters various masks during training, it must learn general relationships between different elements. As a result, MTM may be able to squeeze out more learning signal from any given trajectory.

2.4.7 Representations of MTM

Finally, we study if the representations learned by MTM are useful for downstream learning with traditional RL algorithms. If this is the case, MTM can also be interpreted as an offline

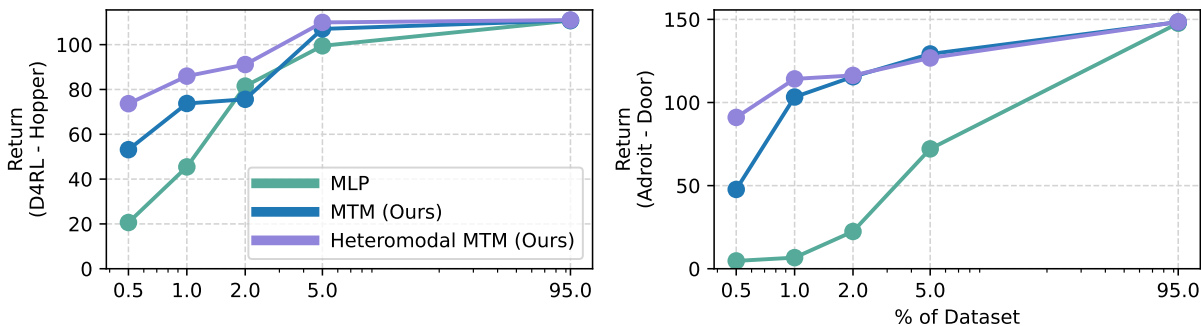


Figure 2.6: **Dataset efficiency.** We train MTM in the D4RL Hopper and Adroit Door environments across a range of dataset sizes, measured by the percent of the original dataset (≈ 1 million transitions). We see that MTM is able to consistently outperform specialized MLP models in the low data regime. Furthermore, we see that Heteromodal MTM (i.e. MTM trained on heteromodal data containing both state-only and state-action trajectories) is further able to provide performance improvement in low data regimes.

pre-training exercise to help downstream RL. To instantiate this in practice, we consider the setting of offline RL using TD3 on the ExORL dataset. The baseline method is to simply run TD3 on this dataset using the raw state as input to the TD3 algorithm. We compare this to our proposed approach of using MTM state representations for TD3. To do this, we first pretrain an MTM model on state-action sequences in the ExORL dataset. Subsequently, to use state representations from MTM, we simply use the MTM encoder to tokenize and encode each state individually. This latent representation of the state can be used in the place of raw states for the TD3 algorithm. The critic of TD3 is conditioned on states and actions. We additionally test state-action representations of MTM by using the latent representation of the state and action encoded jointly with MTM. We allow end to end finetuning of the representations during training. We compare training TD3 on raw states to training TD3 with (a) state representations from the MTM model, and (b) state-action representations from the MTM model with the offline RL loss (i.e. TD3 objective).

Figure 2.7 depicts the learning curves for the aforementioned experiment. In all cases we see significant improvement in training efficiency by using MTM representations – both with state and state-action representations. In the Walk task, we note it actually *improves* over the asymptotic performance of the base TD3 Fujimoto et al. (2018b) algorithm within 10% of training budget. Additionally, we find that the state-action representation from MTM can provide significant benefits, as in the case of the Walk task. Here, finetuning state-action representation from MTM leads to better asymptotic performance compared to state-only representation or learning from scratch. We provide additional plots of MTM frozen representations in Appendix A.5.3

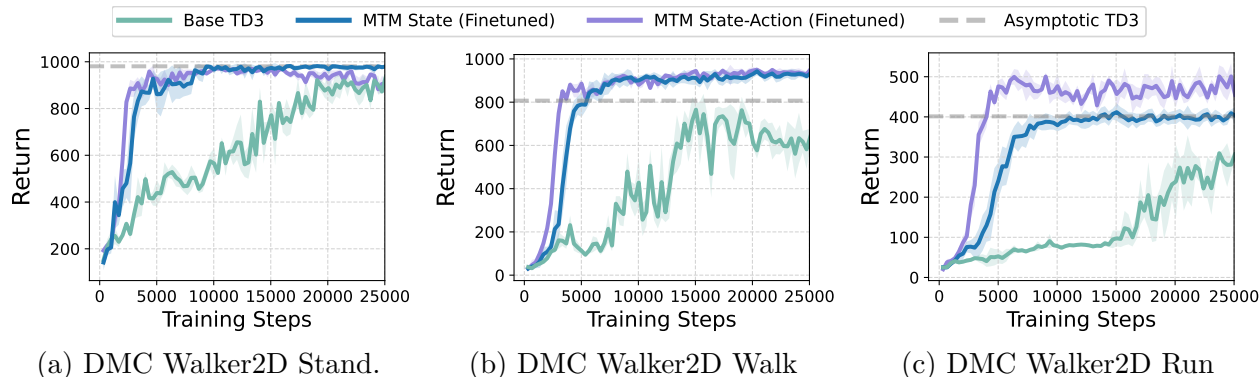


Figure 2.7: **MTM Representations enable faster learning.** The plot visualizes a walker agent’s performance as it is trained using TD3 on different representations across 3 tasks (Stand, Walk, Run). The agent is trained completely offline using data from the ExORL dataset. For MTM state representations, we encode the raw state with MTM. MTM state-action representations additionally jointly encode the state and action for the critic of TD3. The learning curves show that finetuned MTM representations enable the agent to more quickly learn the task at hand, reaching or exceeding the asymptotic performance of TD3 on raw states. Both MTM state representations and MTM state-action representations are comparable in terms of learning speed and performance. In addition, we see that in some cases, like the Run task, state-action representations from MTM help achieve better performance than alternatives. We also show the asymptotic performance reached by TD3 on raw states and actions after training for 100000 iterations and plot the average of 5 seeds.

2.5 Summary

In this chapter, we introduced MTM as a versatile and effective approach for sequential decision making. We empirically evaluated the performance of MTM on a variety of continuous control tasks and found that a single pretrained model (i.e. same weights) can be used for different downstream purposes like inverse dynamics, forward dynamics, imitation learning, offline RL, and representation learning. This is accomplished by simply changing the masks used at inference time. In addition, we showcase how MTM enables training on heterogeneous datasets without any change to the algorithm. Future work includes incorporating training in online learning algorithms for more sample efficient learning, scaling MTM to longer trajectory sequences, and more complex modalities like videos.

Chapter 3

Trajectory Clustering for Imitation Learning

This chapter is based on the paper “Semi-Supervised One-Shot Imitation Learning” (Wu et al., 2024a), by Philipp Wu, Kouros Hakhmaneshi, Yuqing Du, Igor Mordatch, Aravind Rajeswaran, and Pieter Abbeel

One-shot Imitation Learning (OSIL) aims to imbue AI agents with the ability to learn a new task from a single demonstration. To supervise the learning, OSIL typically requires a prohibitively large number of paired expert demonstrations – i.e. trajectories corresponding to different variations of the same semantic task. To overcome this limitation, we introduce the semi-supervised OSIL problem setting, where the learning agent is presented with a large dataset of trajectories with no task labels (i.e. an unpaired dataset), along with a small dataset of multiple demonstrations per semantic task (i.e. a paired dataset). This presents a more realistic and practical embodiment of few-shot learning and requires the agent to effectively leverage weak supervision from a large dataset of trajectories. Subsequently, we develop an algorithm specifically applicable to this semi-supervised OSIL setting. Our approach first learns an embedding space where different tasks cluster uniquely. We utilize this embedding space and the clustering it supports to self-generate pairings between trajectories in the large unpaired dataset. Through empirical results on simulated control tasks, we demonstrate that OSIL models trained on such self-generated pairings are competitive with OSIL models trained with ground-truth labels, presenting a major advancement in the label-efficiency of OSIL.

3.1 Introduction

Humans are capable of learning new tasks and behaviors by imitating others we observe. Furthermore, we are remarkably data efficient, often requiring just a single demonstration. One-shot imitation learning (OSIL) (Duan et al., 2017) aims to imbue AI agents with similar

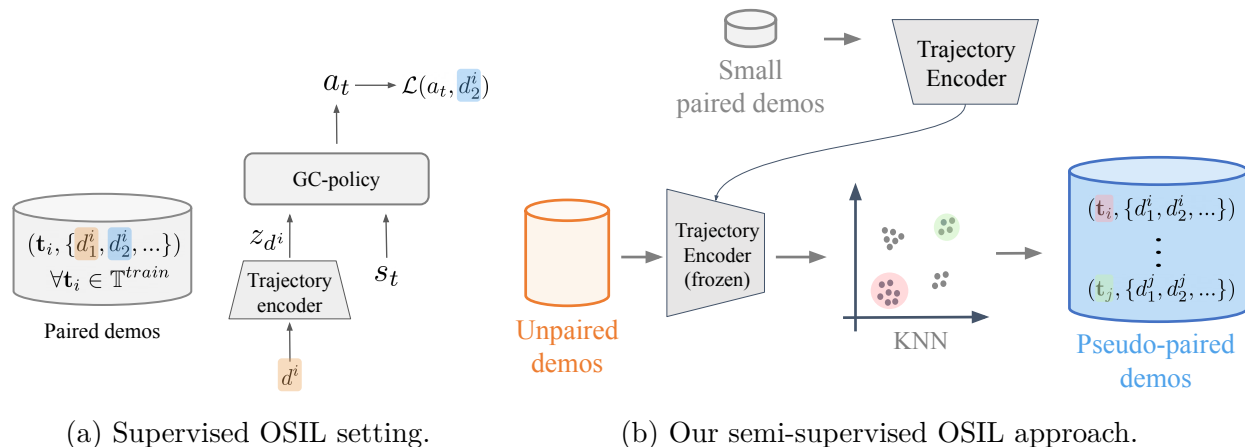


Figure 3.1: (Left) Depiction of the supervised (classical) OSIL setting, where the encoder and policy are trained using several trajectories (d) sharing the same task label (t). (Right) Our semi-supervised OSIL setting instead requires only a large unlabelled dataset of trajectories, and a small paired dataset. For our method, a teacher trajectory encoder is first trained using the labeled dataset. This encoder is then used to construct a pseudo-paired trajectory set by retrieving the k nearest neighbors of each trajectory. We can then train a student on this pseudo-labeled dataset, as in supervised OSIL. Optionally, this relabelling and training procedure can be repeated iteratively.

capabilities. It takes a meta-learning (Schmidhuber, 1987; Naik & Mammone, 1992; Thrun & Pratt, 1998) approach and considers several paired demonstrations – i.e. expert trajectories corresponding to different variations of the semantic task. OSIL learns to reconstruct one trajectory by conditioning on its paired trajectory, implicitly capturing the task semantics. At test time, the resulting agent can directly complete a new task by conditioning on a demonstration of the said task. However, this method often requires prohibitively large amounts of paired trajectories such that the agent experiences enough task variations in diverse environment instantiations to learn a generalizable policy. Collecting such a dataset of demonstrations can be prohibitively expensive, requiring significant engineering effort and/or human data annotation time. In order to improve the data efficiency of OSIL, and expand its applicability, we introduce and study a semi-supervised paradigm for OSIL.

In recent years, we have seen an increase in our ability to collect unsupervised trajectory data in several applications including robotics. This includes access to historical offline datasets (Levine et al., 2020; Fu et al., 2020; Gulcehre et al., 2020), teleoperation and play data in virtual reality (Rajeswaran et al., 2018; Lynch et al., 2019; Gupta et al., 2019), and reward-free exploration (Pathak et al., 2017; Eysenbach et al., 2018; Liu & Abbeel, 2021). Our goal is to leverage these large, abundant, but unlabelled datasets to create a more scalable pathway for OSIL. A direct and naive application of OSIL would require humans to manually annotate these datasets with semantic task descriptions, or manually pair together

similar trajectories, which can be expensive and time consuming. We draw inspiration from semi-supervised learning (van Engelen & Hoos, 2019) in computer vision and natural language processing (NLP), which has emerged as a dominant paradigm to utilize a small labeled dataset in conjunction with large quantities of unlabeled data to train high-quality models (Yang & Yu, 2020; Xie et al., 2020; Chen et al., 2020; Devlin et al., 2018). Analogously, we aim to bring the power of semi-supervised learning to OSIL by learning from both task-agnostic and unlabelled trajectories as well as a small dataset of annotated (paired) trajectories.

Our algorithmic approach to semi-supervised OSIL is based on self-training (Triguero et al., 2013; Yarowsky, 1995; Xie et al., 2020), a prominent approach to semi-supervised learning. In self-training, a teacher network is first trained on a small labeled dataset, and then used to provide pseudo-labels for a larger unlabeled dataset (Hailat & Chen, 2018). This process is repeated multiple times to progressively learn higher quality labels for the entire dataset, ultimately training models with competitive performance despite considerably reduced data annotation effort. To adapt this self-training approach to semi-supervised OSIL, we start with training a teacher encoder-decoder architecture in the standard supervised OSIL fashion, as illustrated in Figure 3.1 (a), with the available paired dataset. We show that even when the teacher does not reach a high task success, the embedding space is sufficiently structured to distinctly cluster different semantic tasks, enabling the generation of pseudo-labelled pairings between nearest neighbors in the embedding space. By bootstrapping on the pseudo-labels obtained from the trajectory clusters in embedding space, we can train a student architecture that outperforms the teacher.

Our Contributions in this work are summarized below.

1. We introduce and formalize the semi-supervised OSIL setting.
2. We propose a novel label-efficient student-teacher trajectory relabeling approach for semi-supervised OSIL that extends the ideas of self-training and distillation from CV and NLP.
3. In a semantic goal navigation task, we find that our method enables an agent trained with only 15% of labelled data to match a fully supervised agent. In a sequential goal navigation task our method approaches fully supervised performance with only 5% of labelled data.
4. We ablate each component of our method, demonstrating their importance to the overall algorithmic contribution.

After the anonymous review phase, we are committed to providing open source for the environments and experiments to facilitate reproducibility and future extensions.

3.2 Related work

One-Shot Imitation Learning (OSIL) The OSIL framework was originally introduced by Duan et al. (2017) to endow AI agents with the capability to learn from a single demonstration.

OSIL relies on access to “paired” demonstrations – i.e. expert trajectories that correspond to different variations of the same semantic task. OSIL then learns by conditioning on one trajectory to reconstruct the paired demonstration, enabling it to implicitly learn the notion of task semantics. Through this view, OSIL has parallels to meta-learning or learning-to-learn (Ren et al., 2018; Finn et al., 2017a; Vinyals et al., 2016; Chebotar et al., 2021; Rajeswaran et al., 2019) as studied broadly in (supervised) machine learning and inverse RL (Das et al., 2020; Yu et al., 2019).

Since the original work of Duan et al. (2017), OSIL has seen several extensions including extensions to visual observation spaces (Finn et al., 2017b), improving task-level generalization (Mandi et al., 2021), and architectural innovations like transformers (Dasari & Gupta, 2021). Nevertheless, the need for a large number of paired demonstrations has limited the broad applicability of OSIL. Our work aims to improve this label efficiency of OSIL by also effectively utilizing a large number of unlabelled (i.e. unpaired) demonstrations, which are often substantially easier to obtain, for example through play data collection (Lynch et al., 2019).

Semi-Supervised Learning The field of semi-supervised learning (Zhu, 2005) studies methods to simultaneously learn from large unlabelled datasets and small labelled datasets. Computer vision, NLP, and speech recognition have been exploring ways to utilize large unlabelled datasets scraped from the internet without expensive and time-intensive human annotations. This has resulted in a wide array of approaches to semi-supervised learning (Zhu, 2005; van Engelen & Hoos, 2019). One dominant paradigm involves pre-training visual representations using unlabelled datasets followed by downstream supervised learning. The representations can be pre-trained with contrastive learning (Hjelm et al., 2019; Chen et al., 2020), generative modeling (Goodfellow et al., 2014), autoencoders (Vincent et al., 2008; He et al., 2021; Wu et al., 2023d) and more. However, such representations lack knowledge of downstream task, and thus might be harder to train, require human priors like appropriate choice of augmentations, or demand very large quantities of unlabelled data.

An alternative and popular approach to semi-supervised learning is self-training (Triguero et al., 2013; Yarowsky, 1995; Xie et al., 2020), where a supervised “teacher” model is first trained on a small labelled dataset and used to generate pseudo-labels for the unsupervised dataset. Subsequently, a student model is trained on both the supervised dataset and the pseudo-labelled dataset. We refer readers to survey works (van Engelen & Hoos, 2019) on semi-supervised learning for more discussion. Our algorithmic approach to semi-supervised OSIL is closer to self-training, and thus has the advantage of being more task-directed in nature. We also perform contrastive representation learning as an auxiliary task and find that it plays an important role, but is insufficient by itself.

Semi-Supervised Learning in RL and IL Improving label efficiency for policy learning, through approaches similar to semi-supervised learning, has been studied in other contexts like reward and goal labels. Prior works tackle the challenge of learning from data without reward

or goal labels by either explicitly learning a reward function through inverse reinforcement learning (Abbeel & Ng, 2004; Ziebart et al., 2008; Finn et al., 2016b), adversarial imitation learning (Ho & Ermon, 2016; Fu et al., 2018a; Rafailov et al., 2021), learning a reward/goal classifier (Fu et al., 2018b; Eysenbach et al., 2021), or by simply assuming a pseudo baseline reward (Yu et al., 2022). In contrast to such prior work, we focus on improving the label efficiency of OSIL, where the need for a large number of paired demonstrations has limited real-world applicability. To our knowledge, our work is the first to study semi-supervised learning approaches to improve label efficiency for OSIL.

3.3 Problem Formulation

Following Duan et al. (2017), in supervised OSIL we denote a set of tasks as \mathbb{T} , each individual task $\mathbf{t} \in \mathbb{T}$, and a distribution of demonstrations of task \mathbf{t} as $\mathbb{D}(\mathbf{t})$. The supervised OSIL objective is to train a policy which, conditioned on a demonstration $d \sim \mathbb{D}(\mathbf{t})$, can accomplish a task \mathbf{t} . This amounts to learning a goal conditioned policy $\pi_\theta(a_t|s_t, d)$, parameterized by θ , that takes an expert demonstration and the current state of the environment as input and emits the proper actions at each time-step t (we differentiate time t and task \mathbf{t} , which is in bold). During training, we have access to a large dataset of demonstrations $d^{\text{train}} \sim \mathbb{D}(\mathbf{t}_i^{\text{train}})$, for a set of training tasks $\mathbf{t}_i^{\text{train}} \in \mathbb{T}^{\text{train}} \subset \mathbb{T}$, where \mathbf{t}_i is the i^{th} task. We formulate the dataset \mathcal{D} as follows

$$\mathcal{D} = \{(\mathbf{t}_i, \{d_1^i, d_2^i, \dots\}) \forall \mathbf{t}_i \in \mathbb{T}^{\text{train}}\}, \quad (3.1)$$

We further assume the existence of a binary valued function $R_{\mathbf{t}}(d)$ which indicates whether a given demonstration or policy rollout d successfully accomplishes the task \mathbf{t} , which we use for evaluating our method. At test time, the policy is provided with one new test demonstration $d^{\text{test}} \sim \mathbb{D}(\mathbf{t})$ that can be either be a new demonstration of a seen task (i.e. $\mathbf{t} \in \mathbb{T}^{\text{train}}$) or a new demonstration of an unseen task (i.e. $\mathbf{t} \in \mathbb{T} \setminus \mathbb{T}^{\text{train}}$).

Semi-supervised OSIL builds on the supervised OSIL setting, which we formulate as follows. We similarly assume access to a small labeled dataset of demonstrations $\mathcal{D}^{\text{labeled}}$ where each demonstration has its associated task label. We additionally assume access to a large dataset of demonstrations $\mathcal{D}^{\text{unlabeled}}$ which does not have the associated task label \mathbf{t}_i . These datasets are defined below:

$$\mathcal{D}^{\text{labeled}} = \{(\mathbf{t}_i, \{d_1^i, d_2^i, \dots\}) \forall \mathbf{t}_i\} \quad (3.2)$$

$$\mathcal{D}^{\text{unlabeled}} = \{d_1, d_2, \dots\} \quad (3.3)$$

An effective semi-supervised method should be able to leverage both annotated and unannotated datasets effectively to maximize the performance of the OSIL agent at test time.

3.4 Method

At its core, OSIL can be simply construed as two modules that are jointly optimized together: (1) an encoder network $f_\phi(d)$ which embeds demonstrated trajectories into a latent space z , and (2) a policy decoder $\pi_\theta(a_t|s_t, z)$ that is conditioned on the demonstration embedding and current state of the environment to output actions. The prior state of the art work on OSIL (Duan et al., 2017; Dasari & Gupta, 2021; Mandi et al., 2021) learn both the demonstration encoder module and the policy decoder jointly by minimizing the predicted action errors on the imitated trajectory, possibly with other auxiliary losses. This method works well when paired trajectories are abundant. In the more realistic semi-supervised OSIL setting, the question becomes “*How can we group sufficiently abundant demonstration pairs from the unlabeled data to train an OSIL agent?*” To address this, we propose an iterative student teacher method.

3.4.1 Student-Teacher Training

The core of our hypothesis is that discriminating or clustering trajectories that share the same semantic task is easier (and thus more data efficient) compared to generative modeling of actions to accomplish a task. To instantiate this in practice, we use a teacher-student self-training paradigm (Xie et al., 2020) to effectively remove the need for large human-annotation on task labels. In our setting, a “teacher” is the encoder f_ϕ that embeds trajectories into the latent space. Using a quality teacher encoder, we can retrieve the k -nearest neighbors of each trajectory in the dataset using a distance measure (e.g L2 distance) on the embedding space and use that as a labeled pair for downstream training of a student OSIL policy.

To train the teacher encoder, we proceed with the standard OSIL training procedure on the smaller labeled dataset, $\mathcal{D}^{labeled}$. The encoder and policy are trained end to end with an imitation loss on the predicted action from the policy, $\pi_\theta(a_t|s_t, z)$, where $z = f_\phi(d_t)$. To encourage learning a more structured latent space, we also employ a contrastive InfoNCE loss (Oord et al., 2018), where a positive pair is taken from the labeled subset of data, and the rest of the goals in the batch are treated as negative examples. This structured latent space is necessary for teacher relabelling. In general, we also find that the contrastive loss helps with learning a better OSIL policy with higher task success rate, which is consistent with the works of James et al. (2018); Mandi et al. (2021).

After training the teacher encoder to convergence, we then generate a set of pseudo labels for the trajectories in the unlabeled dataset. This is done by embedding all of the demonstrations of the dataset $\mathcal{D}^{unlabeled}$ with the teacher encoder f_ϕ . We then find the k nearest neighbors of each demonstration in the embedding space, where k is a hyperparameter. Let $k\text{NN}_\phi(d, \mathcal{D})$ denote the k nearest neighbors of d in the dataset \mathcal{D} using the feature embeddings from a demonstration encoder f_ϕ . If the nearest neighbors are demonstrations associated with the same semantic task, we can supervise an effective student OSIL policy with this

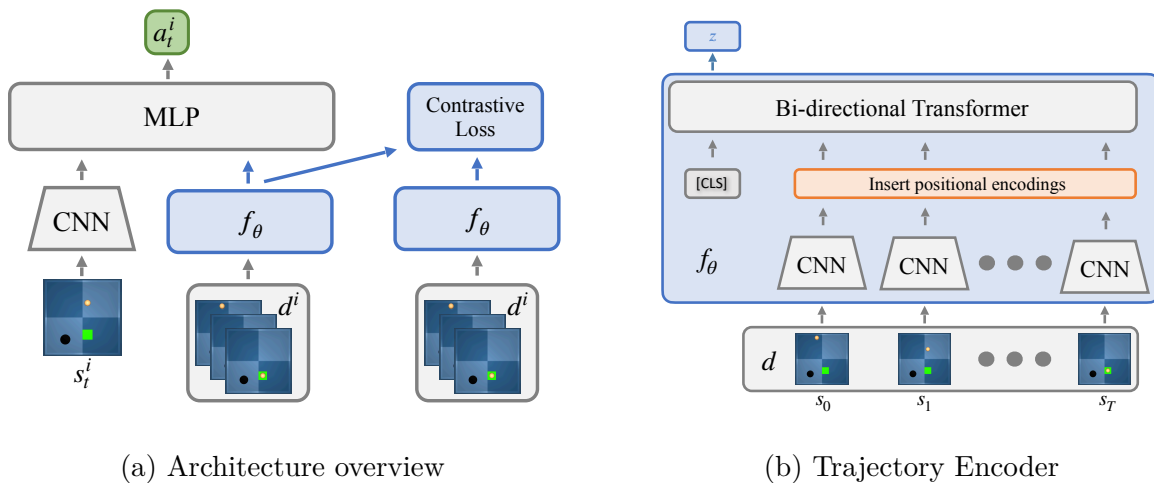


Figure 3.2: The architecture used in our algorithm. (a) shows the generic structure of a OSIL agent, which consists of a generic demonstration encoder f_ϕ and the $\pi_\theta(a_t|s_t, z)$ task latent conditioned policy, which comprises of image encoder.. (b) shows one potential instantiation of the demonstration encoder, which leverages a bi direction transformer to encode the trajectory. This is used for the pinpad sequential navigation task, which requires reasoning over the entire trajectory.

dataset of pseudo-pairs of trajectories, which we formulate as:

$$\mathcal{D}^{pseudo_labeled} = \{(d_i, \{kNN_\phi(d_i, \mathcal{D}^{unlabeled})\}) \forall d_i \in \mathcal{D}^{unlabeled}\} \quad (3.4)$$

Finally, the student policy is trained using both $\mathcal{D}^{pseudo_labeled}$ and $\mathcal{D}^{labeled}$. During training we continue to use the labeled dataset for the imitation and contrastive losses, but additionally sample batches from the pseudo-labeled dataset, which is trained only with the imitation loss. We can continue iterating this process by treating the encoder f_ϕ of the trained student as the teacher for the subsequent round and improving the accuracy on the KNN retrievals from the unlabeled dataset until we get diminishing returns from the process.

3.4.2 Architecture

An overview of the architecture is shown in Figure 3.2a. We use the same architecture for teacher and student with same number of parameters. In general, the demonstration encoder f_ϕ is flexible and can take any form, but should be expressive enough to learn meaningful representations of the demonstration trajectories. Following conditional policies (Jang et al., 2022), we utilize an MLP policy which takes the demonstration embedding z through FiLM conditioning (Perez et al., 2018). The focus of our work is on the procedure of making OSIL more data efficient. We therefore do not consider more complex encoder decoder architectures, for which we refer to prior work. In this work we also focus our experiments on

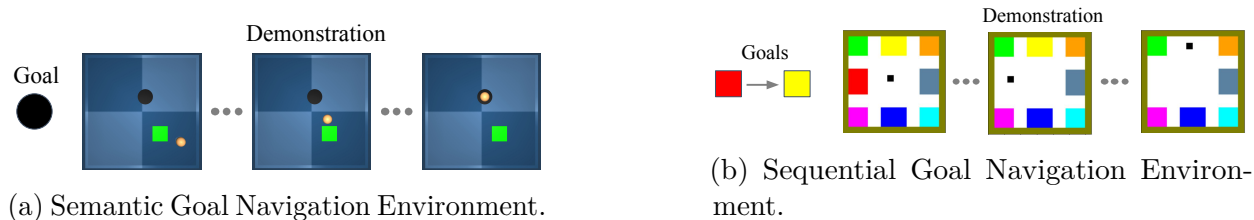


Figure 3.3: Sample goals and corresponding demonstration visualizations for the two tasks.

visual imitation, for which we use a CNN encoder to obtain frame-level visual representations of 64x64 images with a simple 5 layer CNN.

For many OSIL tasks, the final frame is enough to specify the desired intent, which we find true for this environment. A commonly used strategy is to form a summary of the demonstration trajectory by taking a few key frames (Duan et al., 2017; James et al., 2018). For these tasks (e.g. goal reaching) we simply use the final frame image embedding as the representation of the task. Figure 3.2b on the other hand, illustrates a more general solution to embed the entire demonstrated trajectory. In this model, we treat the embedding of each frame as a separate token and use a bi-directional transformer to learn the task encoding. The transformer model has the capacity to learn which frames are important to fully describe the task. Refer to Appendix B.1.1 for more hyperparameter details.

3.5 Experiments

Through our experiments, we aim to study the effectiveness of the semi-supervised OSIL setting, as well as the performance of our algorithm. Concretely, we study the following questions.

- How to train the demonstration encoder to effectively cluster trajectories?
- How to use the learned clusters to effectively improve agent performance?

3.5.1 Environment setup

Semantic Goal Navigation. We construct a custom pointmass-based reaching task using the MuJoCo simulator (Todorov et al., 2012) with the DMControl suite (Tunyasuvunakool et al., 2020). This task is inspired from the simulated reaching task first introduced in (Finn et al., 2017b). The task is to navigate the pointmass to a goal of a given color and shape when also presented with a distractor goal of a different color and shape. Concretely, there are 2 shapes and 5 possible colors the shapes can take on, totalling 10 variations for each object, and 100 possible semantic scenes. See Figure 3.3 for a visual illustration. Note that within each scene configuration, the locations of the objects can be randomized. We collect

800 trajectories for each target goal object, resulting in a total training dataset size of 8000 trajectories.

Sequential Goal Navigation. We use a modified version of the discrete pinpad world environment from Hafner et al. (2022). This task requires the agent to navigate and press two buttons out of six in a specified order. The agent is only considered successful if it is able to correctly reach all the goals in the correct sequence. There are 6 possible goal pads for the agent to reach, totaling 30 tasks. The agent’s action is one of five possible actions: up, down, left, right, or no-op. The observation space is the raw pixels in the environment. See Figure 3.3 for a visual illustration. We randomize both the color assignments of the pads and the agent starting location for each task variation. The agent must pay attention to the entire trajectory to correctly determine the desired task. As such, we parametrize the demonstration encoder for this environment as a small bi-directional transformer that takes in a sequence of states and a class token to predict a latent z encoding of the trajectory.

Dataset Collection. We employ a scripted policy to collect demonstrations for each task variation. Specifically, we reset the initial state of the environment and agent randomly, then run the scripted expert policy. During training we limit the number of demonstrations per each task that the agent gets to see for supervision in order to create a semi-supervised scenario. However, we use the entire collected dataset as a large pool of unlabeled expert trajectories during training. We evaluate our method on two environments described above.

3.5.2 Metrics

Task Success. Our goal is to maximize task success rate using limited task labeled demonstrations. For both environments, we report the success rate of the agent as the performance after 100 trials in the environment, averaged over 3 seeds. We evaluate the agent on both new instantiations of the training tasks and an unseen test task, which we report as "Train" and "Test" respectively. We use different numbers of the total labeled trajectories to show how the number of labeled trajectories effects final task performance.

Trajectory Retrieval (TR) Score For each trajectory in $d^{\text{test}} \in \mathcal{D}$, we retrieve the K nearest neighbors by measuring the L2 distance in the embedding space of the teacher. Let d_i^{ret} be the i^{th} retrieved trajectory and \mathbf{t} be the task label of d^{test} . For each trajectory, the retrieval accuracy is defined as the percentage of time that $R_{\mathbf{t}}(d_i^{\text{ret}}) = 1$. We take an average of this measure across all samples in the training set.

$$TR_{\text{score}}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \frac{1}{k} \sum_{j=1}^k R_{\mathbf{t}}(d_j^{\text{ret}}) \quad (3.5)$$

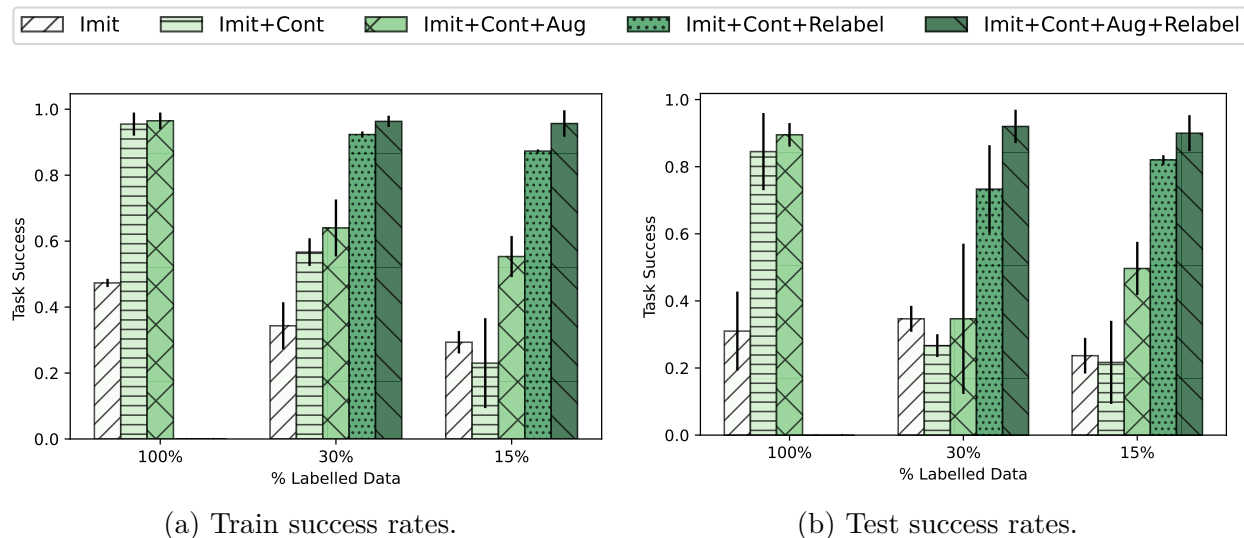


Figure 3.4: Task success rates for the Semantic Goal Navigation Task.

3.5.3 Results

For each experiment, we train the OSIL policy using the learned goal embedding and behavior cloning loss on the labeled subset of data. We report the task success rate and trajectory retrieval scores for all experiments.

Semantic Goal Navigation First we consider the Semantic Goal Navigation pointmass task. We consider 5 main settings:

1. An agent trained with only the imitation loss on the demonstrated actions.
2. An agent trained with an additional contrastive loss on the goal embeddings in addition to to the imitation loss.
3. The same as (2.) but with an added self-supervised loss on the entire dataset (including unlabeled data).
4. A student model trained by using the demonstration encoder (2.) as a teacher model.
5. A student model trained by using the demonstration encoder of (3.) as a teacher model.

The model trained with the method specified in (3.) acts as an alternative semi-supervised baseline in the special case of using the final frame as the demonstration representation. In this setting, we use the supervised labels as in the supervised OSIL case, but further leverage the unlabeled trajectories through adding an additional self supervised loss contrastive loss on augmentations of the goal image (Oord et al., 2018). The augmentations we use are restricted to random flip and random crop.

Table 3.1: Trajectory Retrieval: Final Frame Semantic Goal Navigation

% Labeled Data	Method	Retrieval % with $k=$				
		1	10	50	100	200
100%	Imitation	11.3	11.	10.7	10.6	10.5
	+Contrastive	90.9	91	91.2	90.9	90.7
	+Contrastive+Aug	93.5	92.5	91.7	91.3	90.8
30%	Imitation	11.8	11.8	11.5	11.4	11.3
	+Contrastive	88.8	88.5	88.1	87.90	87.8
	+Contrastive+Aug	93.7	92.9	91.9	91.4	90.8
	+Contrastive+Relabel	91.1	90.6	90.3	89.9	89.6
	+Contrastive+Aug+Relabel	93.7	92.5	91.2	90.7	90.3
15%	Imitation	11.6	11.2	11.0	10.9	10.8
	+Contrastive	63.6	62.8	61.6	61.	60.
	+Contrastive+Aug	91.6	90.6	90.	89.9	89.7
	+Contrastive+Relabel	74.3	73.2	71.9	71.2	70.4
	+Contrastive+Aug+Relabel	91.8	90.7	90.1	89.9	89.6

Figure 3.4 show the task performance across each experiment. As expected, train and validation performance drops when the amount of labeled data decreases. Without relabelling, we see some task performance gains from applying both the contrastive loss and self unsupervised losses. After training a student model using the pseudo-labels from the representations learned by the teacher encoder, we see a leap in performance, matching an agent that has access to 100% ground truth labels, even when only using 15% of the labels.

Table 3.1 shows the trajectory retrieval scores across different values of k . Despite having much less labeled data and decreased task performance, the retrieval scores consistently remain high. This suggests that we are able to learn a meaningful representation for the task, allowing us to cluster the trajectories. We find that the contrastive loss is necessary for learning representations that have high retrieval. Interestingly, we find that relabeling gives much greater gains for task success while augmentation gives more benefit for retrieval, which supports the hypothesis that relabeling gives datapoints for the OSIL training and the contrastive loss (which typically relies on augmented views of data points) helps representation learning, but in a way that is not directly optimizing for the task objective. We show 2D visualizations of the learned embedding in Figure 3.5. We additionally explore the effect of the the number of possible pairs k in Appendix B.1.2.

Sequential Goal Navigation Next we examine a task which requires the trajectory encoder to learn a time-dependent encoding of the trajectory, rather than having the task fully specified by the final frame. For this we employ the more general trajectory demonstration encoder

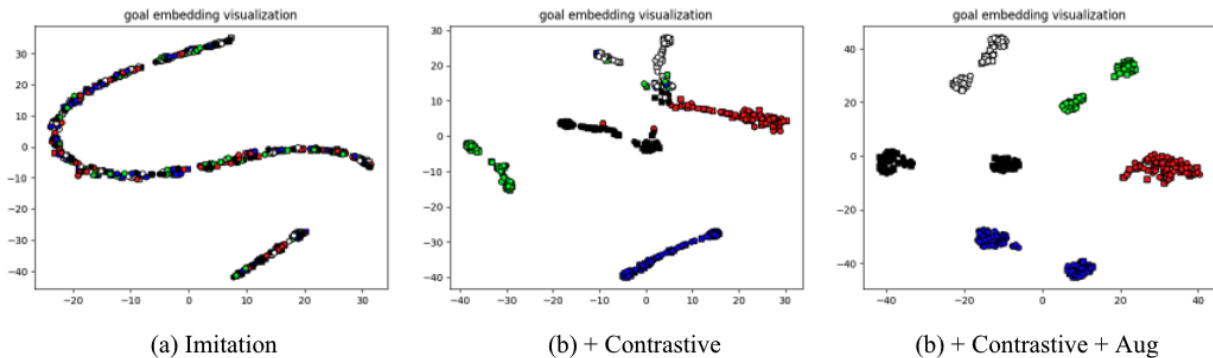


Figure 3.5: TSNE visualizations of the learned embeddings where the only 15% of the dataset is labeled. (a) shows the embedding trained with imitation loss only. (b) adds the contrastive loss on the labeled subset of data (c) additionally adds a self supervised loss with images augmentations.

Table 3.2: Task Success Results: PinPad

% Labeled	Method	Task Success %	
		Train	Val
100%	+Contrastive	92.3 ± 2.9	41.7 ± 15.1
10%	+Contrastive	70.7 ± 4.5	18.3 ± 3.3
	+Contrastive+Relabel	84 ± 4.6	36.3 ± 10.6
5%	+Contrastive	49 ± 1.4	7.7 ± 1.6
	+Contrastive+Relabel	82.3 ± 10	34.3 ± 22.6

shown in Figure 3.2b. Similarly, we see in Table 3.2 that our teacher-student relabeling method allows the agent to improve task performance and almost match the agent trained with a fully labeled dataset, even in much lower labeled data regimes (5%). This suggests that the teacher encoder is able to pay attention to the temporal nature of the demonstrations and generate effective pseudo-labels. Similar to the semantic navigation task, the learned encoder maintains a high trajectory retrieval score across different choices of k .

3.6 Conclusions

In this chapter, we introduce the problem setting of semi-supervised OSIL, which we believe to be a more realistic setting for developing OSIL methods that can scale to real world settings. In semi-supervised OSIL we aim to maximize agent performance in settings where we have access to a large set of task-agnostic expert demonstrations, but only a small task-labeled

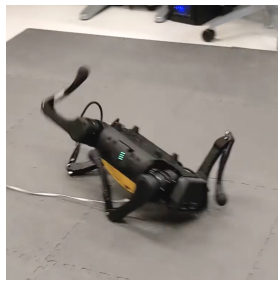
dataset. We introduce a student teacher training method and show that training a teacher network based on the limited labeled data and bootstrapping on the resulting task encoder can allow us to assign effective pseudo-labels to the large unlabeled dataset. Using the pseudo-labeled dataset to train a student network can result in out-performing its teacher, reaching task performance parity with a model trained on much more labeled data. We evaluate our methodology on simulated environments with varying complexity and showed that this can be a promising direction towards semi-supervised OSIL.

Our work aims to provide agents the ability to quickly imitate a demonstration. The work does not assume any particular type of demonstration. A malicious actor might be able to provide nefarious demonstrations to AI agents and safeguards must be considered when deploying such imitation learning systems in the real world. At a more immediate level, we do not anticipate any societal risks due to this work.

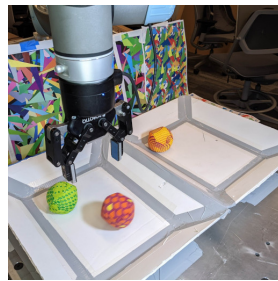
Chapter 4

Robot Learning with World Models

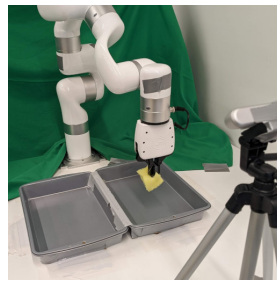
This chapter is based on the paper “DayDreamer: World Models for Physical Robot Learning” (Wu et al., 2022), by Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel.



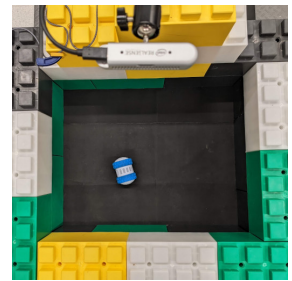
(a) A1 Quadruped Walking



(b) UR5 Visual Pick Place



(c) XArm Visual Pick Place



(d) Sphero Navigation Place

Figure 4.1: To study the applicability of Dreamer for sample-efficient robot learning, we apply the algorithm to learn robot locomotion, manipulation, and navigation tasks from scratch in the real world on 4 robots, without simulators. The tasks evaluate a diverse range of challenges, including continuous and discrete actions, dense and sparse rewards, proprioceptive and camera inputs, as well as sensor fusion of multiple input modalities. Learning successfully using the same hyperparameters across all experiments, Dreamer establishes a strong baseline for real world robot learning.

To solve tasks in complex environments, robots need to learn from experience. Deep reinforcement learning is a common approach to robot learning but requires a large amount of trial and error to learn, limiting its deployment in the physical world. As a consequence, many advances in robot learning rely on simulators. On the other hand, learning inside of simulators fails to capture the complexity of the real world, is prone to simulator inaccuracies,

and the resulting behaviors do not adapt to changes in the world. The Dreamer algorithm has recently shown great promise for learning from small amounts of interaction by planning within a learned world model, outperforming pure reinforcement learning in video games. Learning a world model to predict the outcomes of potential actions enables planning in imagination, reducing the amount of trial and error needed in the real environment. However, it is unknown whether Dreamer can facilitate faster learning on physical robots. In this chapter, we apply Dreamer to 4 robots to learn online and directly in the real world, without any simulators. Dreamer trains a quadruped robot to roll off its back, stand up, and walk from scratch and without resets in only 1 hour. We then push the robot and find that Dreamer adapts within 10 minutes to withstand perturbations or quickly roll over and stand back up. On two different robotic arms, Dreamer learns to pick and place objects from camera images and sparse rewards, approaching human-level teleoperation performance. On a wheeled robot, Dreamer learns to navigate to a goal position purely from camera images, automatically resolving ambiguity about the robot orientation. Using the same hyperparameters across all experiments, we find that Dreamer is capable of online learning in the real world, which establishes a strong baseline. We release our infrastructure for future applications of world models to robot learning. Videos are available on the project website: <https://danijar.com/daydreamer>

4.1 Introduction

Teaching robots to solve complex tasks in the real world is a foundational problem of robotics research. Deep reinforcement learning (RL) offers a popular approach to robot learning that enables robots to improve their behavior over time through trial and error. However, current algorithms require too much interaction with the environment to learn successful behaviors. Recently, modern *world models* have shown great promise for data efficient learning in simulated domains and video games (Hafner et al., 2019a, 2020). Learning world models from past experience enables robots to imagine the future outcomes of potential actions, reducing the amount of trial and error in the real environment needed to learn.

While learning accurate world models can be challenging, they offer compelling properties for robot learning. By predicting future outcomes, world models allow for planning and behavior learning given only small amounts of real world interaction (Gal et al., 2016; Ebert et al., 2018). Moreover, world models summarize general dynamics knowledge about the environment that, once learned, could be reused for a wide range of downstream tasks (Sekar et al., 2020). World models also learn representations that fuse multiple sensor modalities and integrate them into latent states, reducing the need for sophisticated state estimators. Finally, world models generalize well from available offline data (Yu et al., 2021), which further accelerates learning in the real world.

Despite the promises of world models, learning accurate world models for the real world is an open challenge. In this chapter, we leverage recent advances of the Dreamer world model for training a variety of robots in the most straight-forward and fundamental problem

setting: online reinforcement learning in the real world, without simulators or demonstrations. As shown in Figure 4.2, Dreamer learns a world model from a replay buffer of past experience, learns behaviors from rollouts imagined in the latent space of the world model, and continuously interacts with the environment to explore and improve its behaviors. Our aim is to push the limits of robot learning directly in the real world and offer a robust platform to enable future work that develops the benefits of world models for robot learning. The key contributions of this chapter are summarized as follows:

- **Dreamer on Robots** We apply Dreamer to 4 robots, demonstrating successful learning directly in the real world, without introducing new algorithms. The tasks cover a range of challenges, including different action spaces, sensory modalities, and reward structures.
- **Walking in 1 Hour** We teach a quadruped from scratch in the real world to roll off its back, stand up, and walk in only 1 hour. Afterwards, we find that the robot adapts to being pushed within 10 minutes, learning to withstand pushes or quickly roll over and get back on its feet.
- **Visual Pick and Place** We train robotic arms to pick and place objects from sparse rewards, which requires localizing objects from pixels and fusing images with proprioceptive inputs. The learned behavior outperforms model-free agents and approaches the performance of a human teleoperator using the same control interface as the robot.
- **Open Source** We publicly release the software infrastructure for all our experiments, which supports different action spaces and sensory modalities, offering a flexible platform for future research of world models for robot learning in the real world.

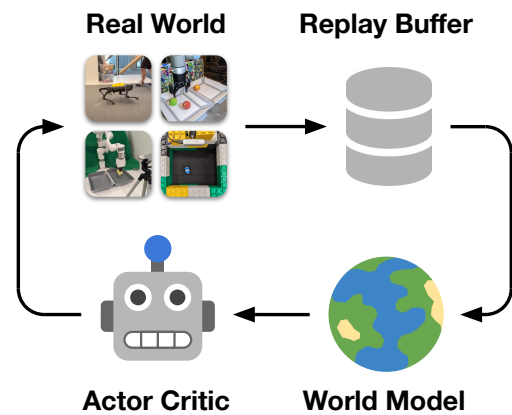


Figure 4.2: Dreamer follows a simple pipeline for online learning on robot hardware without simulators. The current learned policy collects experience on the robot. This experience is added to the replay buffer. The world model is trained on replayed off-policy sequences through supervised learning. An actor critic algorithm optimizes a neural network policy from imagined rollouts in the latent space of the world model. We parallelize data collection and neural network learning.

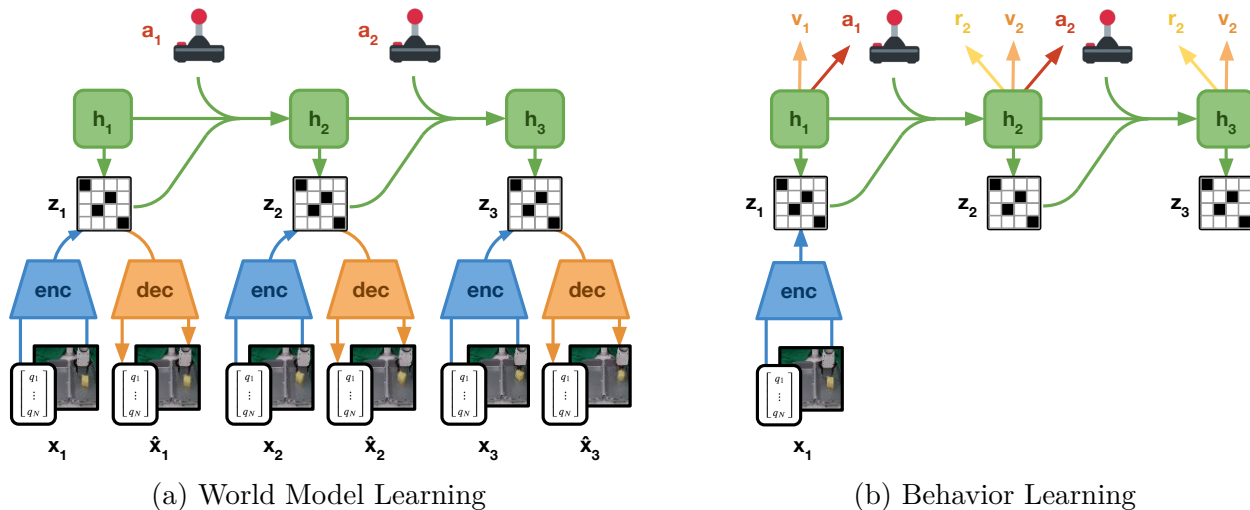


Figure 4.3: **Neural Network Training** We leverage the Dreamer algorithm (Hafner et al., 2019a, 2020) for fast robot learning in real world. Dreamer consists of two main neural network components, the world model and the policy. **Left:** The world model follows the structure of a deep Kalman filter that is trained on subsequences drawn from the replay buffer. The **encoder** fuses all sensory modalities into discrete codes. The decoder reconstructs the inputs from the codes, providing a rich learning signal and enabling human inspection of model predictions. A **recurrent state-space model** (RSSM) is trained to predict future codes given actions, without observing intermediate inputs. **Right:** The world model enables massively parallel policy optimization from imagined rollouts in the compact latent space using a large batch size, without having to reconstruct sensory inputs. Dreamer trains a **policy network** and **value network** from the imagined rollouts and a learned **reward function**.

4.2 Approach

We leverage the Dreamer algorithm (Hafner et al., 2019a, 2020) for online learning on physical robots, without the need for simulators. Figure 4.2 shows an overview of the approach. Dreamer learns a world model from a replay buffer of past experiences, uses an actor critic algorithm to learn behaviors from trajectories predicted by the learned model, and deploys its behavior in the environment to continuously grow the replay buffer. We decouple learning updates from data collection to meet latency requirements and to enable fast training without waiting for the environment. In our implementation, a learner thread continuously trains the world model and actor critic behavior, while an actor thread in parallel computes actions for environment interaction.

World Model Learning The world model is a deep neural network that learns to predict the environment dynamics, as shown in Figure 4.3 (left). Because sensory inputs can be

large images, we predict future representations rather than future inputs. This reduces accumulating errors and enables massively parallel training with a large batch size. Thus, the world model can be thought of as a fast simulator of the environment that the robot learns autonomously, starting from a blank slate and continuously improving its model as it explores the real world. The world model is based on the Recurrent State-Space Model (RSSM; Hafner et al., 2019b), which consists of four components:

$$\begin{aligned} \text{Encoder Network: } & \text{enc}_\theta(s_t | s_{t-1}, a_{t-1}, x_t) & \text{Decoder Network: } & \text{dec}_\theta(s_t) \approx x_t \\ \text{Dynamics Network: } & \text{dyn}_\theta(s_t | s_{t-1}, a_{t-1}) & \text{Reward Network: } & \text{rew}_\theta(s_{t+1}) \approx r_t \end{aligned} \quad (4.1)$$

Physical robots are often equipped with multiple sensors of different modalities, such as proprioceptive joint readings, force sensors, and high-dimensional inputs such as RGB and depth camera images. The encoder network fuses all sensory inputs x_t together into the stochastic representations z_t . The dynamics model learns to predict the sequence of stochastic representations by using its recurrent state h_t . The decoder reconstructs the sensory inputs to provide a rich signal for learning representations and enables human inspection of model predictions. In our experiments, the robot has to discover task rewards by interacting with the real world, which the reward network learns to predict. Using manually specified rewards as a function of the decoded sensory inputs is also possible. We optimize all components of the world model jointly by stochastic backpropagation (Kingma & Welling, 2013; Rezende et al., 2014).

Actor Critic Learning While the world model represents task-agnostic knowledge about the dynamics, the actor critic algorithm learns a behavior that is specific to the task at hand. As shown in Figure 4.3 (right), we learn behaviors from rollouts that are predicted in the latent space of the world model, without decoding observations. This enables massively parallel behavior learning with typical batch sizes of 16K on a single GPU. The actor critic algorithm consists of an actor network $\pi(a_t | s_t)$ and a critic network $v(s_t)$.

The role of the actor network is to learn a distribution over successful actions a_t for each latent model state s_t that maximizes the sum of future predicted task rewards. The critic network learns to predict the sum of future task rewards through temporal difference learning (Sutton & Barto, 2018). This allows the algorithm to take into account rewards beyond the planning horizon of $H = 16$ steps to learn long-term strategies. Given a predicted trajectory of model states, the critic is trained to regress the return of the trajectory. We compute λ -returns following Hafner et al. (2020, 2019a):

$$V_t^\lambda \doteq r_t + \gamma \left((1 - \lambda)v(s_{t+1}) + \lambda V_{t+1}^\lambda \right), \quad V_H^\lambda \doteq v(s_H). \quad (4.2)$$

While the critic network is trained to regress the λ -returns, the actor network is trained to maximize them. Different gradient estimators are available for computing the policy gradient for optimizing the actor, such as Reinforce (Williams, 1992) and the reparameterization trick (Kingma & Welling, 2013; Rezende et al., 2014) that directly backpropagates return gradients through the differentiable dynamics network (Henaff et al., 2019). Following Hafner et al.

(2020), we choose reparameterization gradients for continuous control tasks and Reinforce gradients for tasks with discrete actions. In addition to maximizing returns, the actor is also incentivized to maintain high entropy to prevent collapse to a deterministic policy and maintain some amount of exploration throughout training:

$$\mathcal{L}(\pi) \doteq -\mathbb{E}\left[\sum_{t=1}^H \ln \pi(a_t \mid s_t) \text{sg}(V_t^\lambda - v(s_t)) + \eta \text{H}[\pi(a_t \mid s_t)]\right] \quad (4.3)$$

We optimize the actor and critic using the Adam optimizer (Kingma & Ba, 2014). To compute the λ -returns, we use a slowly updated copy of the critic network as common in the literature (Mnih et al., 2015; Lillicrap et al., 2015). The actor and critic gradients do not affect the world model, as this would lead to incorrect and overly optimistic model predictions. The hyperparameters are listed in Appendix C.5.

4.3 Experiments

We evaluate Dreamer on 4 robots, each with a different task, and compare its performance to appropriate algorithmic and human baselines. The experiments are representative of common robotic tasks, such as locomotion, manipulation, and navigation. The tasks pose a diverse range of challenges, including continuous and discrete actions, dense and sparse rewards, proprioceptive and image observations, and sensor fusion. The goal of the experiments is to evaluate whether the recent successes of learned world models enables sample-efficient robot learning directly in the real world. Specifically, we aim to answer the following research questions:

- Does Dreamer enable robot learning directly in the real world, without simulators?
- Does Dreamer succeed across various robot platforms, sensory modalities, and action spaces?
- How does the data-efficiency of Dreamer compare to previous reinforcement learning algorithms?

Implementation We build on the official implementation of DreamerV2 (Hafner et al., 2020). We develop an asynchronous actor and learner setup, which is essential in environments with high control rates, such as the quadruped, and also accelerates learning for slower environments, such as the robot arms. The actor thread computes online actions for the robot and sends trajectories of 128 time steps to the replay buffer. The learner thread samples data from the replay buffer, updates the world model, and optimizes the policy using imagination rollouts. Policy weights are synced from the learner to the actor every 20 seconds. We use an RSSM with 256 units to speed up the training computation. We use identical hyperparameters across all experiments, enabling off-the-shelf training on different robot embodiments.

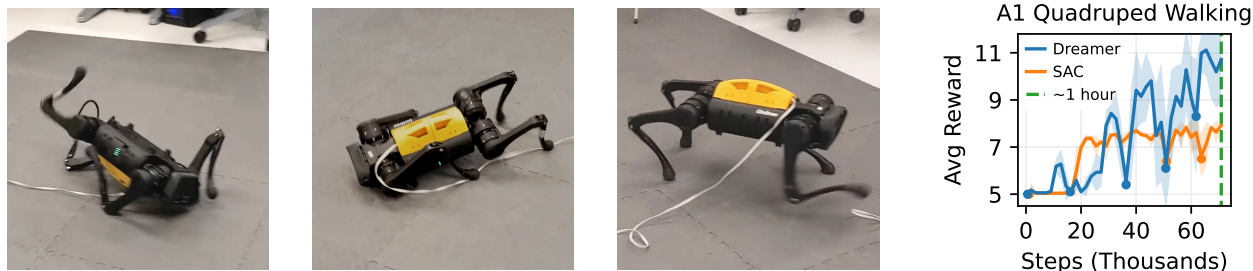


Figure 4.4: **A1 Quadruped Walking** Starting from lying on its back with the feet in the air, Dreamer learns to roll over, stand up, and walk in 1 hour of real world training time, without simulators or resets. In contrast, SAC only learns to roll over but neither to stand up nor to walk. For SAC, we also had to help the robot out of a dead-locked leg configuration during training. On the right we show training curves for both SAC and Dreamer. The maximum reward is 14. The filled circles indicate times where the robot fell on its back, requiring the learning of a robust strategy for getting back up. After 1 hour of training, we start pushing the robot and find that it adapts its behavior within 10 minutes to withstand light pushes and quickly roll back on its feet for hard pushes. The graph shows a single training run with the shaded area indicating one standard deviation within each time bin.

Baselines We compare to a strong learning algorithm for each of our experimental setups. The A1 quadruped robot uses continuous actions and low-dimensional inputs, allowing us to compare to SAC (Haarnoja et al., 2018a,b), a popular algorithm for data-efficient continuous control. For the visual pick and place experiments on the XArm and UR5 robots, inputs are images and proprioceptive readings and actions are discrete, suggesting algorithms from the DQN (Mnih et al., 2015) line of work as baselines. We choose Rainbow (Hessel et al., 2018) as a powerful representative of this category, an algorithm that combines many improvements of DQN. To input the proprioceptive readings, we concatenate them as broadcasted planes to the RGB channels of the image, a common practice in the literature (Schrittwieser et al., 2019). For the UR5, we additionally compare against PPO (Schulman et al., 2017), with similar modifications for fusing image and proprioceptive readings. In addition, we compare against a human operator controlling the robot arm through the robot control interface. For the Sphero navigation task, inputs are images and actions are continuous. The state-of-the-art baseline in this category is DrQv2 (Yarats et al., 2021a), which uses image augmentation to increase sample-efficiency.

4.3.1 A1 Quadruped Walking

This high-dimensional continuous control task requires training a quadruped robot to roll over from its back, stand up, and walk forward at a fixed target velocity. Prior work in quadruped locomotion requires either extensive training in simulation under domain randomization,

using recovery controllers to avoid unsafe states, or defining the action space as parameterized trajectory generators that restrict the space of motions (Rusu et al., 2016; Peng et al., 2018; Rudin et al., 2021; Lee et al., 2020; Yang et al., 2019). In contrast, we train in the end-to-end reinforcement learning setting directly on the robot, without simulators or resets. We use the Unitree A1 robot that consists of 12 direct drive motors. The motors are controlled at 20 Hz via continuous actions that represent motor angles that are realized by a PD controller on the hardware. Actions are filtered with a Butterworth filter to protect the motor from high-frequency actions. The input consists of motor angles, orientations, and angular velocities. Due to space constraints, we manually intervene when the robot has reached the end of the available training area, without modifying the joint configuration or orientation that the robot is in.

The reward function is the sum of five terms. An upright reward is computed from the base frame up vector \hat{z}^T , terms for matching the standing pose are computed from the joint angles of the hips, shoulders, and knees, and a forward velocity term is computed from the projected forward velocity \mathcal{B}_v^x and the total velocity \mathcal{B}_v . Without the reward curriculum, the agent receives spurious reward values due to the velocity estimator’s dependence on foot-ground contact events. Each of the five terms is active while its preceding terms are satisfied to at least 0.7 and otherwise set to 0:

$$\begin{aligned} r^{\text{upr}} &\doteq (\hat{z}^T[0, 0, 1] - 1)/2 & r^{\text{hip}} &\doteq 1 - \frac{1}{4} \|q^{\text{hip}} + 0.2\|_1 & r^{\text{shoulder}} &\doteq 1 - \frac{1}{4} \|q^{\text{shoulder}} + 0.2\|_1 \\ r^{\text{knee}} &\doteq 1 - \frac{1}{4} \|q^{\text{knee}} - 1.0\|_1 & r^{\text{velocity}} &\doteq 5 \left(\max(0, \mathcal{B}_v^x) / \|\mathcal{B}_v\|_2 \cdot \text{clip}(\mathcal{B}_v^x / 0.3, -1, 1) + 1 \right) \end{aligned} \quad (4.4)$$

As shown in Figure 4.4, after one hour of training, Dreamer learns to consistently flip the robot over from its back, stand up, and walk forward. In the first 5 minutes of training, the robot manages to roll off its back and land on its feet. 20 minutes later, it learns how to stand up on its feet. About 1 hour into training, the robot learns a pronking gait to walk forward at the desired velocity. After succeeding at this task, we tested the robustness of the algorithms by repeatedly knocking the robot off of its feet with a large pole, shown in Figure 4.5. Within 10 minutes of additional online learning, the robot adapts and withstand pushes or quickly rolls back on its feet. In comparison, SAC quickly learns to roll off its back but fails to stand up or walk given the small data budget.



Figure 4.5: Within 10 minutes of perturbing the learned walking behavior, the robot adapts to withstand pushes or quickly rolling over and back on its feet.

4.3.2 UR5 Multi-Object Visual Pick and Place

Common in warehouse and logistics environments, pick and place tasks require a robot manipulator to transport items from one bin into another. Figure 4.6 shows a successful



Figure 4.6: **UR5 Multi Object Visual Pick and Place** This task requires learning to locate three ball objects from third-person camera images, grasp them, and move them into the other bin. The arm is free to move within and above the bins and sparse rewards are given for grasping a ball and for dropping it in the opposite bin. The environment requires the world model to learn multi-object dynamics in the real world and the sparse reward structure poses a challenge for policy optimization. Dreamer overcomes the challenges of visual localization and sparse rewards on this task, learning a successful strategy within a few hours of autonomous operation.

pick and place cycle of this task. The task is challenging because of sparse rewards, the need to infer object positions from pixels, and the challenging dynamics of multiple moving objects. The sensory inputs consist of proprioceptive readings (joint angles, gripper position, end effector Cartesian position) and a 3rd person RGB image of the scene. Successfully grasping one of the 3 objects, detected by partial gripper closure, results in a +1 reward, releasing the object in the same bin gives a -1 reward, and placing in the opposite bin gives a +10 reward. We control the UR5 robot from Universal Robotics at 2 Hz. Actions are discrete for moving the end effector in increments along X, Y, and Z axes and for toggling the gripper state. Movement in the Z axis is only enabled while holding an object and the gripper automatically opens once above the correct bin. We estimate human teleoperation performance by recording 3 demonstrators for 20 minutes each, controlling the UR5 with a joystick.

Dreamer reaches an average pick rate of 2.5 objects per minute within 8 hours. The robot initially struggles to learn as the reward signal is very sparse, but begins to gradually improve after 2 hours of training. The robot first learns to localize the objects and toggles the gripper when near an object. Over time, grasping becomes precise and the robot learns to push objects out of corners. Figure 4.6 shows the learning curves of Dreamer compared to Rainbow DQN, PPO, and the human baseline. Both Rainbow DQN and PPO only learn the short-sighted behavior of grasping and immediately dropping objects in the same bin. In contrast, Dreamer approaches human-level teleoperation performance after 8 hours. We hypothesize that Rainbow DQN and PPO fail because they require larger amounts of experience, which is not feasible for us to collect in the real world.

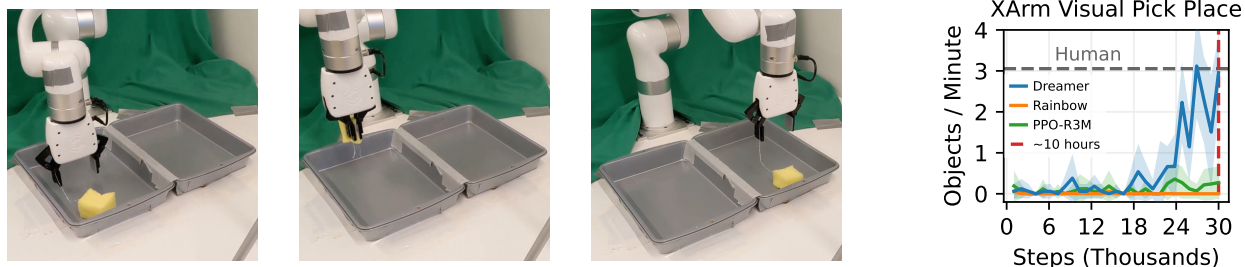


Figure 4.7: **XArm Visual Pick and Place** The XArm is an affordable robot arm that operates slower than the UR5. To demonstrate successful learning on this robot, we use a third-person RealSense camera with RGB and depth modalities, as well as proprioceptive inputs for the robot arm, requiring the world model to learn sensor fusion. The pick and place task uses a soft object. While soft objects would be challenging to model accurately in a simulator, Dreamer avoids this issue by directly learning on the real robot without a simulator. While Rainbow and PPO using R3M visual embeddings converge to the local optimum of grasping and ungrasping the object in the same bin, Dreamer learns a successful pick and place policy from sparse rewards in under 10 hours.

4.3.3 XArm Visual Pick and Place

While the UR5 robot is a high performance industrial robot, the XArm is an accessible low-cost 7 DOF manipulation, which we control at approximately 0.5 Hz. Similar to Section 4.3.2, the task requires localizing and grasping a soft object and moving it from one bin to another and back, shown in Figure 4.7. We connect the object to the gripper with a string, which makes it less likely for the object to get stuck in corners at the cost of more complex dynamics. The sparse reward, discrete action space, and observation space match the UR5 setup except for the addition of depth image observations.

Dreamer learns a policy that enables the XArm to achieve an average pick rate of 3.1 objects per minute in 10 hours of time, which is comparable to human performance on this task. Figure 4.7 shows that Dreamer learns to solve the task within 10 hours, whereas the Rainbow algorithm, a top model-free algorithm for discrete control from pixels, fails to learn. We additionally compare Dreamer against a PPO baseline that utilizes R3M (Nair et al., 2022) pretrained visual embeddings for the state, but notice no improvement in performance. Interestingly, we observed that Dreamer learns to sometimes use the string to pull the object out of a corner before grasping it, demonstrating multi-modal behaviors. Moreover, we observed that when lighting conditions change drastically (such as sharp shadows during sunrise), performance initially collapses but Dreamer then adapts to the changing conditions and exceeds its previous performance after a few hours of additional training, reported in Appendix C.1.

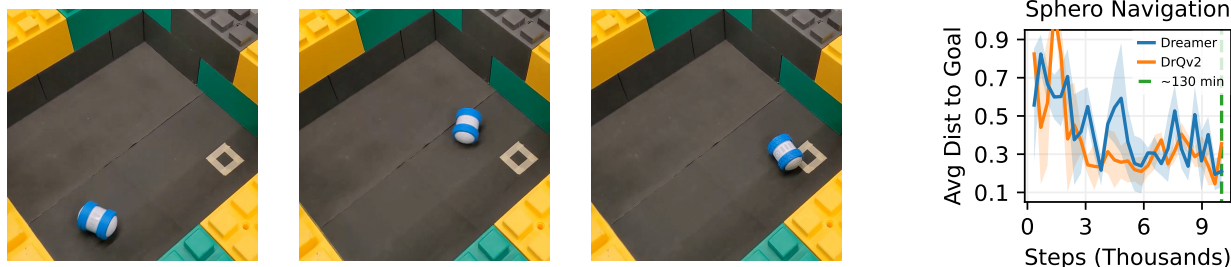


Figure 4.8: **Sphero Navigation** This task requires the Sphero robot to navigate to a goal location given a top-down RGB image as the only input. The task requires the robot to localize itself from raw pixels, to infer its orientation from the sequence of past images because it is ambiguous from a single image, and to control the robot from under-actuated motors that require building up momentum over time. Dreamer learns a successful policy on this task in under 2 hours.

4.3.4 Sphero Navigation

We evaluate Dreamer on a visual navigation task that requires maneuvering a wheeled robot to a fixed goal location given only RGB images as input. We use the Sphero Ollie robot, a cylindrical robot with two controllable motors, which we control through continuous torque commands at 2 Hz. Because the robot is symmetric and the robot only has access to image observations, it has to infer the heading direction from the history of observations. The robot is provided with a dense reward equal to the negative L2 distance, which is computed using a oracle vision pipeline that detects the Sphero’s position (this information is not provided to the agent). As the goal is fixed, after 100 environment steps, we end the episode and randomize the robot’s position through a sequence of high power random motor actions.

In 2 hours, Dreamer learns to quickly and consistently navigate to the goal and stay near the goal for the remainder of the episode. As shown in Figure 4.8, Dreamer achieves an average distance to the goal of 0.15, measured in units of the area size and averaged across time steps. We find that DrQv2, a model-free algorithm specifically designed to continuous control from pixels, achieves similar performance. This result matches the simulated experiments of Yarats et al. (2021a) that showed the two algorithms to perform similarly for continuous control tasks from images.

4.4 Related Work

Existing work on robot learning commonly leverages large amounts of simulated experience before deploying to the real world (Rusu et al., 2016; Peng et al., 2018; OpenAI et al., 2018; Lee et al., 2020; Irpan et al., 2020; Kumar et al., 2021; Siekmann et al., 2021; Escontrela et al., 2022), leverage fleets of robots to collect experience datasets (Kalashnikov et al., 2018; Dasari

et al., 2019; Kalashnikov et al., 2021; Ebert et al., 2021), or rely on external information such as human expert demonstrations or task priors to achieve sample-efficient learning (Xie et al., 2019; Schoettler et al., 2019; James et al., 2021; Shah & Levine, 2022; Bohez et al., 2022; Sivakumar et al., 2022). However, designing simulated tasks and collecting expert demonstrations is time-consuming. Moreover, many of these approaches require specialized algorithms for leveraging offline experience, demonstrations, or simulator inaccuracies. In contrast, our experiments show that learning end-to-end from rewards in the physical world is feasible for a diverse range of tasks through world models.

Relatively few works have demonstrated end-to-end learning from scratch in the physical world. Visual Foresight (Finn et al., 2016a; Finn & Levine, 2017; Ebert et al., 2018) learns a video prediction model to solve real world tasks by online planning, but is limited to short-horizon tasks and requires generating images during planning, making it computationally expensive. Yang et al. (2019, 2022b) learn quadruped locomotion through a model-based approach by predicting foot placement and leveraging a domain-specific controller to achieve them. Ha et al. (2020b) learn a quadruped walking policy by relying on a scripted reset policy, so the robot does not have to learn to stand up. SOLAR (Zhang et al., 2019) learns a latent dynamics model from images and demonstrates reaching and pushing with a robot arm. Nagabandi et al. (2019) learns manipulation policies by planning through a learned dynamics model from state observations. In comparison, our experiments show successful learning across 4 challenging robot tasks that cover a wide range of challenges and sensory modalities, with a single learning algorithm and hyperparameter setting.

4.5 Discussion

We applied Dreamer to physical robot learning, finding that modern world models enable sample-efficient robot learning for a range of tasks, from scratch in the real world and without simulators. We also find that the approach is generally applicable in that it can solve robot locomotion, manipulation, and navigation tasks without changing hyperparameters. Dreamer taught a quadruped robot to roll off the back, stand up, and walk in 1 hour from scratch, which previously required extensive training in simulation followed by transfer to the real world or parameterized trajectory generators and given reset policies. We also demonstrate learning to pick and place objects from pixels and sparse rewards on two robot arms in 8–10 hours.

Limitations While Dreamer shows promising results, learning on hardware over many hours creates wear on robots that may require human intervention or repair. Additionally, more work is required to explore the limits of Dreamer and our baselines by training for a longer time. Finally, we see tackling more challenging tasks, potentially by combining the benefits of fast real world learning with those of simulators, as an impactful future research direction.

Acknowledgements We thank Stephen James and Justin Kerr for helpful suggestions and help with printing the protective shell of the quadruped robot. We thank Ademi Adeniji for help with setting up the XArm robot and Raven Huang for help with setting up the UR5 robot. This work was supported in part by an NSF Fellowship, NSF NRI #2024675, and the Vanier Canada Graduate Scholarship.

Chapter 5

Trajectory Data Collection for Manipulation

This chapter is based on the paper “GELLO: A General, Low-Cost, and Intuitive Teleoperation Framework for Robot Manipulators” (Wu et al., 2023e), by Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel.

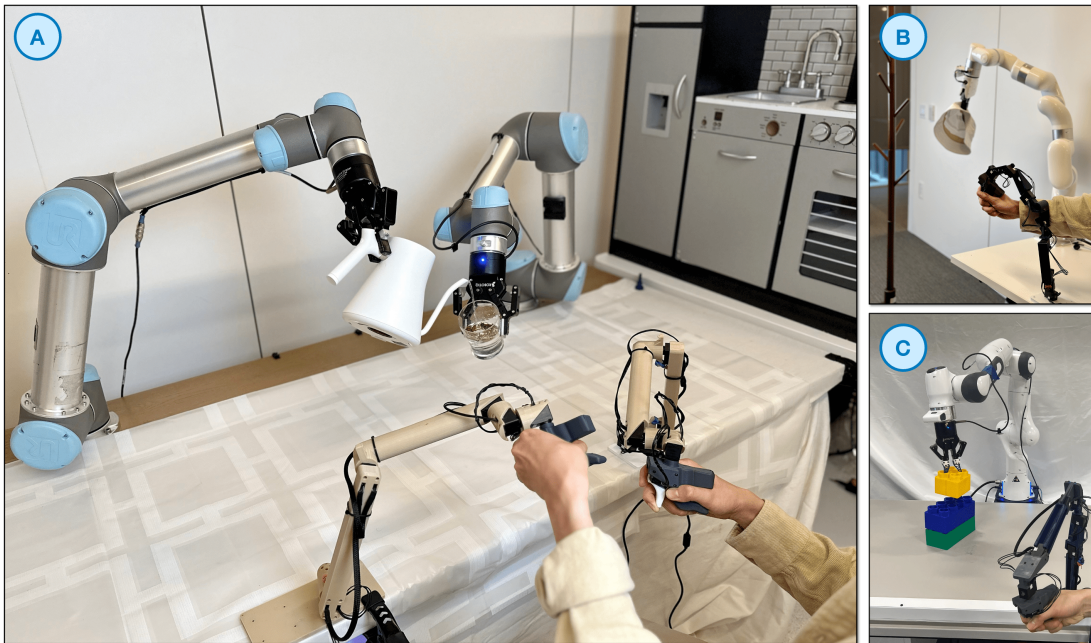


Figure 5.1: We show GELLO teleoperation systems built for three different types of robots: (A) two UR5s, (B) an xArm, and (C) a Franka. The user can teleoperate the robot arms by controlling the GELLO devices. The bill of materials for each GELLO device is less than \$300.

Humans can teleoperate robots to accomplish complex manipulation tasks. Imitation learning has emerged as a powerful framework that leverages human teleoperated demonstrations to teach robots new skills. However, the performance of the learned policies is bottlenecked by the quality, scale, and variety of the demonstration data. In this chapter, we aim to lower the barrier to collecting large and high-quality human demonstration data by proposing a GEneral framework for building LOw-cost and intuitive teleoperation systems for robotic manipulation (GELLO). Given a target robot arm, we build a GELLO controller device that has the same kinematic structure as the target arm, leveraging 3D-printed parts and economical off-the-shelf motors. GELLO is easy to build and intuitive to use. Through an extensive user study, we show that GELLO enables more reliable and efficient demonstration collection compared to other cost efficient teleoperation devices commonly used in the imitation learning literature such as virtual reality controllers and 3D spacemouses. We further demonstrate the capabilities of GELLO for performing complex bi-manual and contact-rich manipulation tasks. To make GELLO accessible to everyone, we have designed and built GELLO systems for 3 commonly used robotic arms: Franka, UR5, and xArm. All software and hardware are open-sourced and can be found on our website: <https://wuphilipp.github.io/gello/>.

5.1 Introduction

In recent years, robotics has gone through a remarkable transformation driven by the increasing integration of data-driven methods in every component, ranging from perception to control. The ability to learn from diverse data helps robots generalize to new scenarios which would be difficult to achieve by a manually designed system. For manipulation, there have been great successes in leveraging imitation learning (Schaal, 1999; Argall et al., 2009; Siciliano & Khatib, 2007; Zhang et al., 2018) where system performance improves with larger datasets (Brohan et al., 2022, 2023; Jang et al., 2021; Reed et al., 2022). However, existing systems are still bottlenecked by the dataset size as well as the complexity and diversity of the tasks within the dataset.

Human teleoperation has consistently proven to be one of the best ways to control robot manipulators to collect demonstrations for imitation learning. Teleoperation has a long history and enabled impressive robot capabilities in a diverse range of challenging situations such as fine grain manipulation of everyday tasks (Zhao et al., 2023), robot surgery (Guo et al., 2019a; Pugin et al., 2011; Liang et al., 2017), underwater exploration (Brantner & Khatib, 2021; Barbieri et al., 2018), and disaster relief (Wang et al., 2015; Katyal et al., 2014), all of which would be useful skills for a robot to learn. A common feature of the teleoperation devices used in these systems is their bilateral capability, which enables the operator to not only direct commands at the robot but also receive force feedback from the robot’s environment. While performant, these teleoperation systems typically require high-fidelity sensors due to their strict system requirements, leading to higher costs that ultimately limit accessibility.

As a result, a more commonly used approach is to build a teleoperation system that captures control signals from lower-cost commodity electronic devices like 3D-mouse (Connection, 2023), VR controllers (Meta, 2024; Vive, 2023) or cameras, which is then converted to a robot action. However, these systems abstract away the kinematic constraints of the robot and can be unintuitive to new users. “A Low-cost Open-source Hardware System for Bimanual Teleoperation” (ALOHA) presents a teleoperation system leveraging an off-the-shelf servo-based arm to control a manipulator of similar size and kinematics, showing impressive teleoperation capabilities for fine-grained manipulation tasks despite being unilateral (Zhao et al., 2023). Nevertheless, the ALOHA system is tailored to a specific robot arm and has a higher cost due to having additional robot arms as controllers for the user.

In this chapter, we introduce GELLO, a **GE**neral, **LO**w-cost, and intuitive teleoperation framework for robot manipulators, targeted towards enabling a scalable way to collect demos by exploring the limits of affordability and simplicity. GELLO is designed to be low-cost, easy to build, and intuitive for humans to use. The key principle is to build miniature, kinematically equivalent controllers with 3D-printed parts and off-the-shelf motors as joint encoders. We clarify that the ideas behind GELLO are not new, rather our contributions can be summarized in the three points below:

1. We present practical implementations of GELLO as a teleoperation system for three commonly used robot arms with simple and low-cost designs.
2. We perform a comprehensive user study demonstrating the system’s effectiveness compared to other prevalent low-cost teleoperation systems in the literature.
3. We fully open-source the hardware and software needed to replicate and operate GELLO to ensure accessibility for the research community to build upon, complete with a bill of materials and assembly instructions¹.

5.2 Related Work

5.2.1 Teleoperation Systems for Manipulation

Low-cost Controllers. Teleoperation systems have a long-standing history and various low-cost sensors have been used to provide the interface for human-robot interaction. Commonly used teleoperation systems include joysticks and spacemouses (Liu et al., 2023c; Lin et al., 2024; Zhu et al., 2022), commercial VR controllers (Zhang et al., 2018; Rakita et al., 2017; Shridhar et al., 2023; Arunachalam et al., 2023), RGB cameras (Handa et al., 2020; Sivakumar et al., 2022; Qin et al., 2023; Song et al., 2020) or IMU sensors (Laghi et al., 2018; Kim et al., 2010; Wu et al., 2019). However due to the morphological differences between these control devices and the robots, the user often can only perform teleoperation in the more abstracted end-effector space. The kinematic constraints of the robot arms therefore are not perceived

¹Website <https://wuphilipp.github.io/gello/>

by the human operators. This prevents the operator from precisely controlling the arm near the areas of kinematic singularities and self-collisions, which reduces the demonstration throughput and increases failures. Moreover, both VR and camera-based solutions can suffer from occlusion and additional latency.

Notably, the recent ALOHA system showcases impressive fine-grained bi-manual manipulation with Dynamixel-based servo arms, where an additional two arms of similar form factor function as controllers (Zhao et al., 2023). Similar to other more conventional but costly teleoperation systems (Hulin et al., 2011; Katz, 2018b; Schwarz et al., 2021), the teleoperation device is another fully-fledged robot arm, with size and capability comparable to the manipulator arm, which can increase the cost. In comparison, we use low-cost components to design a scaled replica of the target arm, resulting in an economical solution that still maintains the advantages of using a kinematically isomorphic arm as the controller.

Bilateral Teleoperation Systems. In contrast to unilateral teleoperation approaches, bilateral teleoperation enables the user to feel force feedback from the target arm (Lawrence, 1993; Hannaford, 1989), an active area of research with a wide range of methodologies. One such approach uses additional robot arms that are isomorphic to the target robots serving as controllers (Hulin et al., 2011; Katz, 2018b; Schwarz et al., 2021; Elsner et al., 2022; Lenz & Behnke, 2021; Whitney et al., 2014). This approach enables environment feedback from the manipulator to be directly relayed back to the joints of the control device. Also, the user can easily sense the kinematic constraints of the robot arm, as the controller arm has the same kinematic constraints. Additionally, there have been efforts to design 1-to-1 exoskeletons for teleoperation purposes (Jo et al., 2013; Toedtheide et al., 2023; Ishiguro et al., 2020). These systems, though effective, are typically bespoke for specific robots, leading to a varied design approach across different robot types. Another avenue explored in the literature is the use of special input devices with haptic feedback (Lab, 2023; Dimension, 2023). While these devices offer a tangible sense of the robot’s kinematic constraints, they often have a very tight operation space and additionally require translating the robot’s kinematic constraints into tangible force feedback increasing system complexity (Zhu et al., 2020; Brantner & Khatib, 2021). In contrast to these approaches, our contribution presents a generalized framework for designing affordable and easily accessible exoskeleton-like unilateral controllers. We provide instances of our approach for three widely-used robot arms. Our system, GELLO, stands out for its affordability, portability, and replicability, reducing the challenges associated with collecting quality teleoperated human demonstrations.

5.2.2 Learning from Human Demonstrations

Learning from demonstrations has been a popular framework for enabling robots to perform a wide range of tasks (Brohan et al., 2022, 2023; Jang et al., 2021; Zhang et al., 2018; Song et al., 2020; Pastor et al., 2009; Bousmalis et al., 2023). Prior works have observed that the performance of the learning system scales with the size of the dataset. As such, there are substantial ongoing efforts at collecting larger and larger datasets (Walke et al., 2023; Fang et al., 2023b). However, collecting human demonstrations can be expensive and

Teleop Device	Approximate Cost
3D Mouse (SpaceNavigator(Connexion, 2023))	\$150
GELLO (Ours)	\$300
VR (Meta Quest 2) (Meta, 2024)	\$300
Robot-to-robot Teleop (e.g. UR5)	\$30,000
Haptic Device (Omega7 (Dimension, 2023))	\$40,000

Table 5.1: An approximate cost comparison of the price of commonly used teleoperation systems for robot learning research. In our user study, we show that GELLO compares favorably to other low-cost options (*e.g.* spacemouse and VR) while being orders of magnitudes cheaper than other options.

time-consuming. For example, the data collection process as done by Brohan et al. (2022) spanned over 17 months with a team of researchers. On the other hand, significant efforts have been made towards better human-robot interaction to address the bottleneck. Some examples include sharing control between the human and the robot (Liu et al., 2023c), or enabling a human to operate multiple robots simultaneously (Dass et al., 2023). These approaches are complementary to our objective, which is to build teleoperation systems that are more accessible and intuitive to use. Finally, a promising direction is learning directly from human videos (Finn et al., 2017b; Smith et al., 2019; Bahl et al., 2022; Qin et al., 2022; Wang et al., 2023a; Wen et al., 2023). While collecting videos of humans performing the tasks directly is relatively inexpensive, overcoming the morphology gap between robots and humans remains challenging.

5.3 Teleoperation Device Design

The focus for the design of GELLO is to create an interface that is both economically accessible and easy-to-use for users aiming to render high-quality demonstrations in robot learning. The primary design principles are summarized as follows:

- **Low-cost:** We aim to show that a capable system can be constructed at an affordable price, thus minimizing the entry barrier. This is achieved through the use of economical backdrivable servo motors, 3D printed components, and a minimalist design, making it possible to construct a teleoperation solution for under \$300. A cost comparison is shown in Table 5.1. We will show that GELLO outperforms other low-cost options in our user study while being much cheaper than the other systems.
- **Capable:** GELLO is designed to be easy to use for human operators. We demonstrate GELLO’s capabilities on a range of complex bi-manual manipulation and contact-rich tasks.

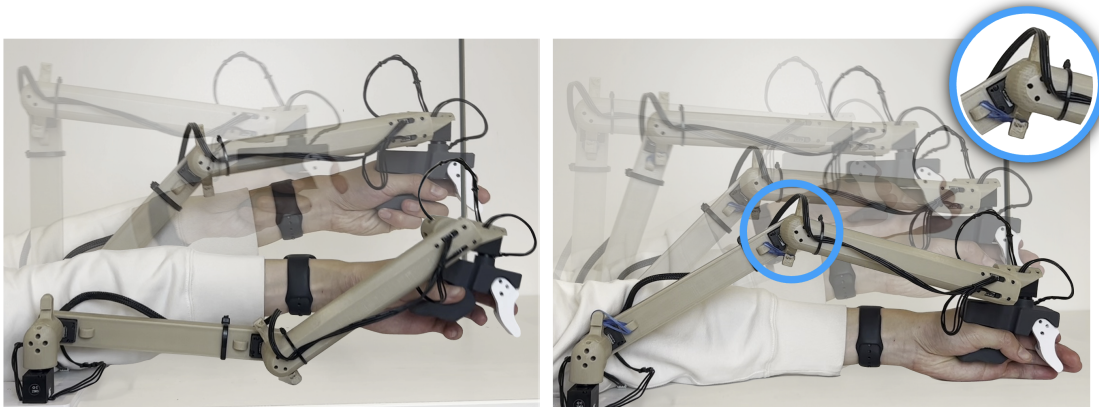


Figure 5.2: This figure illustrates the trajectory of GELLO both with and without joint regularization. Left: Without joint regularization the elbow joint drops. This results in an unfavorable joint configuration for subsequent tasks or even collision with the table. Right: With joint regularization in place, the elbow exhibits minimal movement, leading to a more advantageous joint configuration. We find that simple rubber bands or springs are effective.

- **Portable:** The diversity of demonstration data collected in different tasks and environments is critical to the final performance of the learning system. As such, we design GELLO to be compact, and self-contained, facilitating easy transportation. We show GELLO performing tasks across both lab and in-the-wild environments.
- **Simple to replicate:** The sourcing for parts is minimal beyond off-the-shelf motors and 3D printing components. The assembly process is also straightforward, requiring minimal technical expertise.

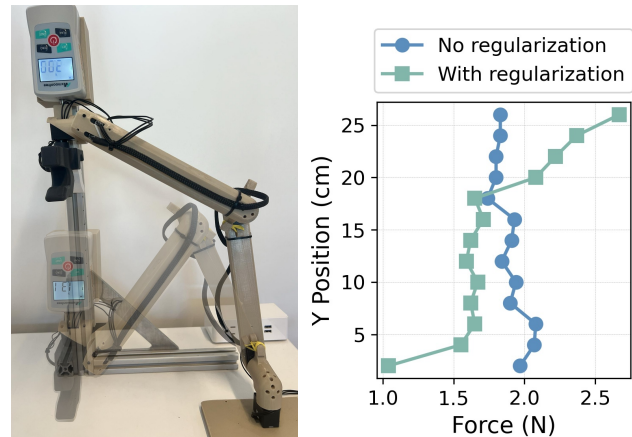
We follow these design principles to make GELLO for three types of robot arms, as shown in Figure 5.1. The instantiation of our approach centers around critical components like motor selection, kinematically equivalent structure, joint regularization, and 3D printed parts.

Servo selection The critical component to enable GELLO’s construction is the availability of low-cost, fully-featured servos. Specifically, we used the DYNAMIXEL XL330 series (ROBOTIS, 2023). Despite their affordability, these servos are equipped with high-resolution 12-bit encoders, enabling joint measurements within 0.088 of a mechanical degree. These encoders provide measurements of the servo’s position, allowing accurate mapping of the controller’s configuration to the target arm. In principle, a servo is not even necessary for the construction of GELLO, as we only need to read joint positions. However, in practice, a servo package provides an easy off-the-shelf, self-contained solution that has an encoder and communication protocol, simplifying construction, usage, and maintenance, furthering the goal of easy replication. In addition, the servo actuator provides physical resistance as the

user backdrives it, which acts as natural damping and improves stability for the user. For this reason, we use the XL-330-288T, which provides the highest gear ratio offering the most resistance.

A scaled kinematically equivalent structure We build GELLO as a small-scale version of the target arm which possesses a kinematically equivalent structure. This means that the joints and links of GELLO correspond directly to those of the target arm, allowing the user to control the GELLO manipulator as if they were directly controlling the target arm, as in kinesthetic teaching (Billard et al., 2006). The kinematically equivalent structure is generated by taking the DH parameters of the target arm, and then scaling the lengths by a factor of α . Although the optimal α for comfort will be user and robot dependent, we used $\alpha = 0.5$ in our implementation and found it to be effective.

Joint positions are read from the GELLO device and directly sent as a joint command to the target arm for operation, avoiding the need to compute inverse kinematics. The user can feel resistance from the controller when the joints are close to kinematic singularities or joint limits and is thus more aware of these failures, leading to more reliable teleoperation. At the same time, the miniature design makes the controller more portable while still allowing the user to operate full-scale robot arms.



Joint regularization With only passive servo motors, the arm is constantly dragged by gravity to undesirable configurations during operation. We find that by adding simple joint regularizers, we can counteract the force of gravity on the manipulator, making it easier for the user to control. We employ rudimentary but effective passive joint regularization that involves the use of mechanical components such as springs or rubber bands to ensure that the device maintains a “natural” posture. This prevents the arm from adopting other kinematically viable yet unconventional positions, as illustrated in Figure 5.2, which may result in collisions. We only add joint regularization elements to the

Figure 5.3: A quantitative analysis of the effects of implementing simple joint regularization on the force necessary to maintain the teleoperation device in a stationary position at various heights. Without the joint regularization, the measured force required to resist gravity is approximately constant around 1.9N. This can make it more difficult for the operator to feel the kinematic limits of the robot. With regularization, we see that near the table, the restorative force helps reduce the required force to compensate for gravity while when far away, a higher restorative force is provided, encouraging the user to avoid singularities.

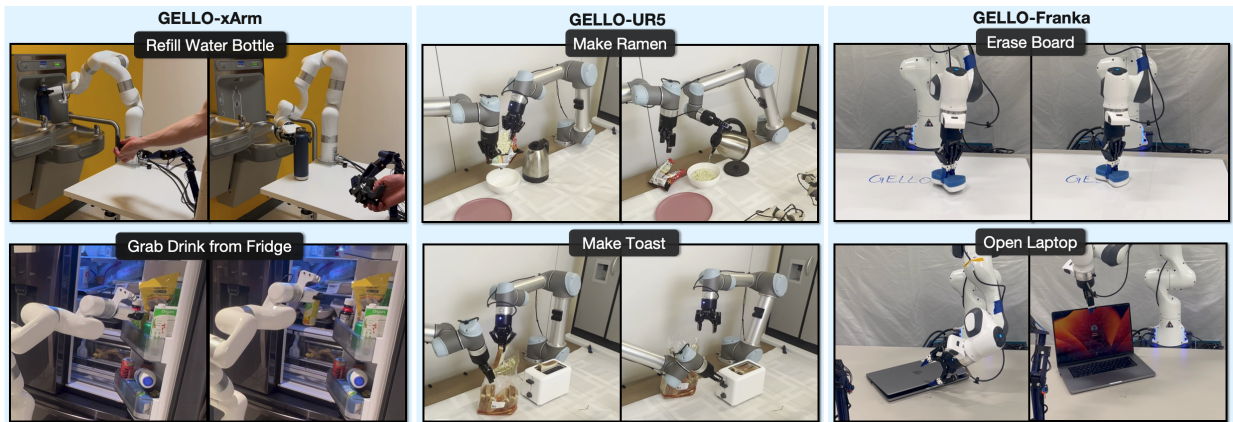


Figure 5.4: We design and instantiate GELLO for 3 different robot arms. We experiment with GELLO across these three arms on a range of tasks to qualitatively observe the teleportation behavior. GELLO’s portable and self-sufficient design enables us to gather data across a wide range of environments.

joints that exhibit the most significant resistance against gravity in the arm’s default resting position, which for the UR design, is the second and third joint. We quantitatively study this in Figure 5.3. We also find that joint regularization provides passive force feedback to the user which differs near the extremities of the joint range. This helps inform the user about the current configuration of the arm.

3D printed parts The use of 3D printed parts in GELLO allows a high degree of customization, enabling users to design and print parts that match the specific robot hardware. 3D printing allows us to easily design GELLO systems for 3 kinematically different robots. 3D printing is also a cost-effective method of producing parts, further contributing to the low-cost nature of GELLO.

Following these simple design principles, we instantiate and test GELLO for three commonly used robot arms, the Universal Robot UR5, uFactory xArm7, and Franka Panda. Example tasks that we can perform with GELLO on the different robots are illustrated in Figure 5.4.

The control setting of direct joint control results in a very simple software stack. Joint angles are read directly from the GELLO device using the DYNAMIXEL provided python API and are commanded to the follower robot. We use the various python APIs for each robot type to send commands to the follower robot. We use ZMQ (ZeroMQ, 2024) for message passing between processes, and provide a simple protocol for extending to new robot types.

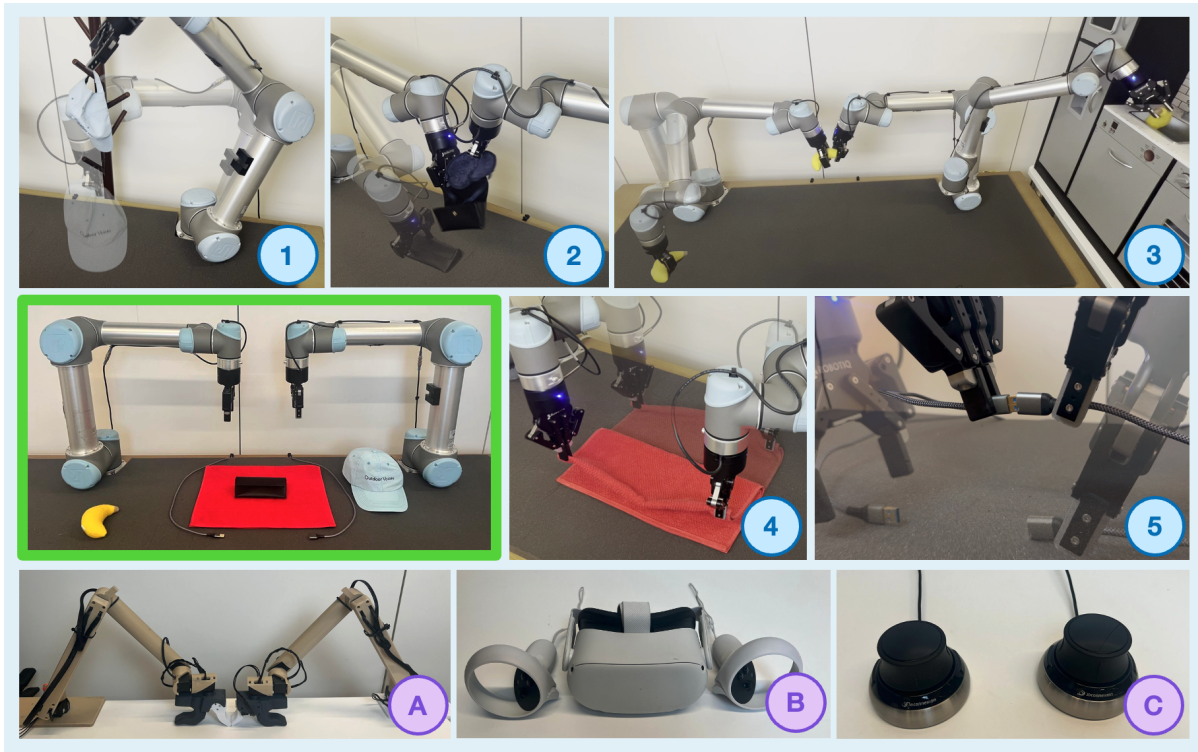


Figure 5.5: An illustration of the experimental setup for our user study. The primary workspace of our experiments, indicated in green, showcases the scene of a bimanual robot station comprising of two UR5 robots with the complete task suit. The figure details the five tasks which are represented by the numerically labelled frames in blue: (1) Place a hat on a rack, (2) Open a case and fetch the sleeping mask inside, (3) Hand over a banana to the kitchen area, (4) Fold a towel, and (5) Plug in a USB cable. These tasks are designed to explore different teleoperation challenges such as articulated object interaction, large workspaces, deformable objects, and precise insertion. The three different bimanual teleoperation devices are shown by the letter-labelled images: (A) GELLO, (B) VR (Meta Quest2) controllers, and (C) 3D Mice (SpaceMouse). Each user attempts to accomplish the 5 tasks with all 3 teleoperation devices.

5.4 Experiments

We conduct experiments to evaluate GELLO as a teleoperation system. Quantitatively, we compare the performance of GELLO with other common low-cost teleoperation systems used in the literature across 5 tasks exploring various aspects of manipulation. Qualitatively, we study GELLO through tests on robots from 3 different manufacturers across a variety of manipulation tasks in diverse settings.

5.4.1 User Study Procedure

We conducted a user study involving 12 participants, focusing on bi-manual robot teleoperation using two UR robots to assess the comparative effectiveness of GELLO, 3D mice, and VR controllers under controlled conditions. All participants were volunteers from our university and none of the participants had professional training in robot teleoperation prior to the study. An overview of our experimental study is shown in Figure 5.5, where details of the 5 tasks we study are provided. These tasks cover a wide range of different manipulation challenges such as control over large work spaces, deformable objects, and fine-grain manipulation. Control with a 3D mouse or VR controller requires additional tuning of gain parameters that affect how sensitive the controller is to human input. A larger gain results in a larger robot motion for a fixed human control input. We tune each by testing the device across the 5 tasks, ensuring control is sensitive enough to achieve the fine-grain USB insertion tasks while still being responsive enough for quickly traversing across the workspace in the banana handoff task.

Before experimenting with the teleoperation tools, each user was granted a brief 6-minute orientation session, introducing them to the basics of the robot and teleoperation itself, as well as the task requirements. To reduce potential biases, no device-specific instructions were given in this orientation, and video demonstrations of the task do not show any particular device. This is then followed by the sequential introduction of all 3 different teleoperation devices. Upon introducing a new device, users were given a 5-minute practice phase, allowing them to gain familiarity with the device and its usage. During this time, users are allowed to experiment with the device and practice the tasks however they please. Next, the participants begin the task execution phase, with a time limit of 45 seconds for the hat task, and 90 seconds for the remaining tasks. This is repeated for all 3 teleoperation devices. *The order in which each participant learns to use the three devices is randomized.* This eliminates any potential bias from a fixed ordering. There are 6 possible orderings that the user can operate the devices. Each order is seen twice. Users were instructed to solve the task as best they could while avoiding self-collision (collision between the follower robot with itself or the other follower robot) or collision with the environment. The robot will be stopped when failure happens and the current task is terminated immediately. We record the task success and failure mode or task completion time as applicable. In the end, we have the user accomplish the same tasks without a robot using their hands. For privacy, no other user-specific data was collected.

5.4.2 User Study Results

In Table 5.2, we show the success rate across distinct tasks for each teleoperation device. Using GELLO consistently results in the top success rate. For the simplest task, which only requires controlling a single arm, placing the hat on the rack, GELLO and VR perform comparably with respect to mean completion time. In more complex tasks, such as the banana handoff which requires both arms to maneuver across the large workspace, using

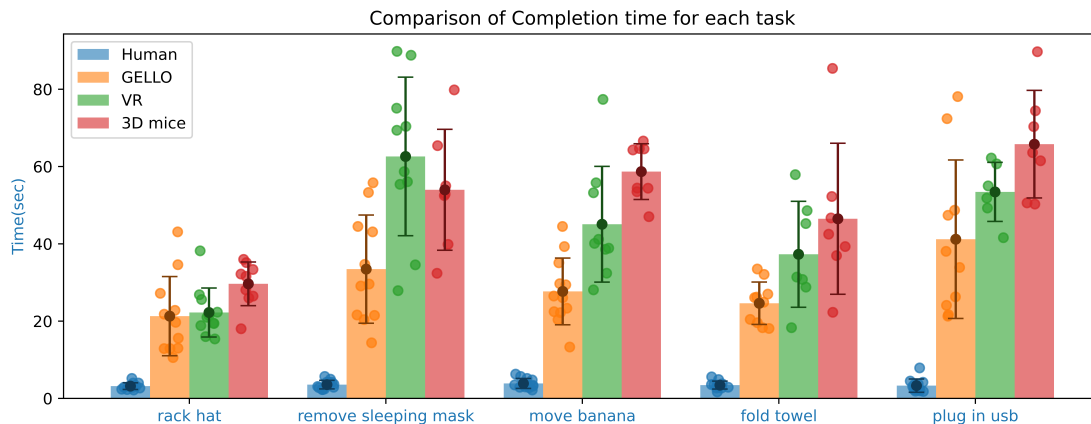


Figure 5.6: Comparison with other teleoperation systems on the required duration for each task. For every combination of task and system, the average completion time (smaller is better) is plotted only for successful trials. Colored dots indicates the completion time for each user under each task-system pairing. Human, plotted in blue, gives a lower bound on teleoperation completion time, where the user directly accomplished each task with their hands.

Table 5.2: The task success rate for different teleoperation systems. GELLO achieves the top success rate across the board.

Device	Hat	Mask	Banana	Towel	USB	Avg
Gello	0.92	0.92	1.0	0.92	0.83	0.92
3D Mice	0.75	0.58	0.67	0.58	0.58	0.63
VR	0.92	0.83	0.75	0.58	0.5	0.72

spacemouse or VR leads to more failures like self-collisions or hitting singularities, suggesting that GELLO offers easier control in more broad use cases. For the task of towel folding, GELLO also shows a large advantage over the other two devices. We hypothesize that this is because GELLO offers more intuitive control and thus better coordination between the two arms. Interestingly, for the task of plugging in the USB cable, using GELLO also resulted in a much higher success rate. This is despite VR or spacemouse having the advantage of simplifying the problem by keeping one controller static while focusing their attention on the other, a common strategy users employed.

In Table 5.3, we present a breakdown on the different failure modes for each teleoperation system. Our observations suggest that, unlike other devices that operate in Cartesian space, GELLO requires minimal user expertise. This is further corroborated by the lowest timeout count for GELLO in comparison to other teleoperation systems. Furthermore, GELLO’s isomorphic joint structure design ensures that teleoperation with GELLO has the least

Table 5.3: The failure count for each failure mode aggregated across all 5 tasks. Each **X** indicates a single failure of that type from our trials. The “Other” category captures all other irrecoverable task failure modes such as dropping the working item outside the reach of the robots.

Failure Mode	GELLO	3D Mice	VR
Timeout	X	XXXXXXXXXX	XXXXX
Self Collision	X	XXXXX	XXXXX
Env Collision	XXX	XXXXXX	XXXXXX
Other		XX	X

collision risk. Qualitatively we observed that self collisions were a common problem for the spacemouse and VR controllers due to the teleoperator paying attention only to the end effector and its relative motion. These devices, unlike GELLO, are unobservant of kinematic and robot-to-robot constraints which could explain the high amount of self-collisions.

Figure 5.6 provides a summary of the completion times taken by each device across all five tasks, given successful task execution. Utilizing GELLO results in consistently faster completion times. This not only signifies that GELLO is easier to use with a higher success rate but also indicates its efficiency; faster completion times would enable users to achieve more successful operations in a given time frame.

5.4.3 Teleoperation System Capabilities

We further demonstrate the capabilities of GELLO on more challenging manipulation tasks, including in real-world environments across the three different robot platforms. We show some of them in Figure 5.4 and put more videos on our project website. These tasks include contact-rich tasks, long-horizon tasks, and challenging bi-manual coordination tasks. Tasks like filling water bottles require a certain payload on the robot arm and would be difficult for smaller arms to perform, such as the ViperX arm used in the ALOHA system (Zhao et al., 2023). GELLO is also effective for 7 DOF arms, such as the Panda and xArms, despite the extra degree of freedom. We find that the kinematically equivalent structure enables a user to directly manage the arm’s null space if required, which can be advantageous when operating in cluttered spaces.

5.5 Discussion

Due to limited output torque of the motors used, GELLO does not provide force feedback to the users, which limits GELLO’s capabilities when teleoperating for more contact-rich tasks.

We made this compromise to keep GELLO low-cost, more accessible, and more applicable to all robot arms, as bilateral devices also require force sensing capability for the target robot. However, we hope to incorporate this as an optional capability in the future for more advanced usage.

Our user study is limited to inexperienced users who are only briefly taught about teleoperation and who only practiced for a limited time. Additional training can significantly improve the user’s proficiency in using teleoperation devices and we leave such study to future work.

In this chapter, we introduced GELLO, a general low-cost teleoperation platform for manipulation. Our results demonstrate its effectiveness through a user study on teleoperation with a bi-manual robot system using two UR5s. To demonstrate versatility and make GELLO more accessible, we design GELLO for 3 robots. We hope GELLO will lower the barrier to collecting large and high-quality demonstration datasets, and thus accelerate progress in robot learning.

Chapter 6

Robot Policy Learning with Human-in-the-Loop

This chapter is based on the paper “” (Wu et al., 2024b), by Philipp Wu, Yide Shentu, Qiayuan Liao, Ding Jin, Menglong Guo, Koushil Sreenath, Xingyu Lin, and Pieter Abbeel.

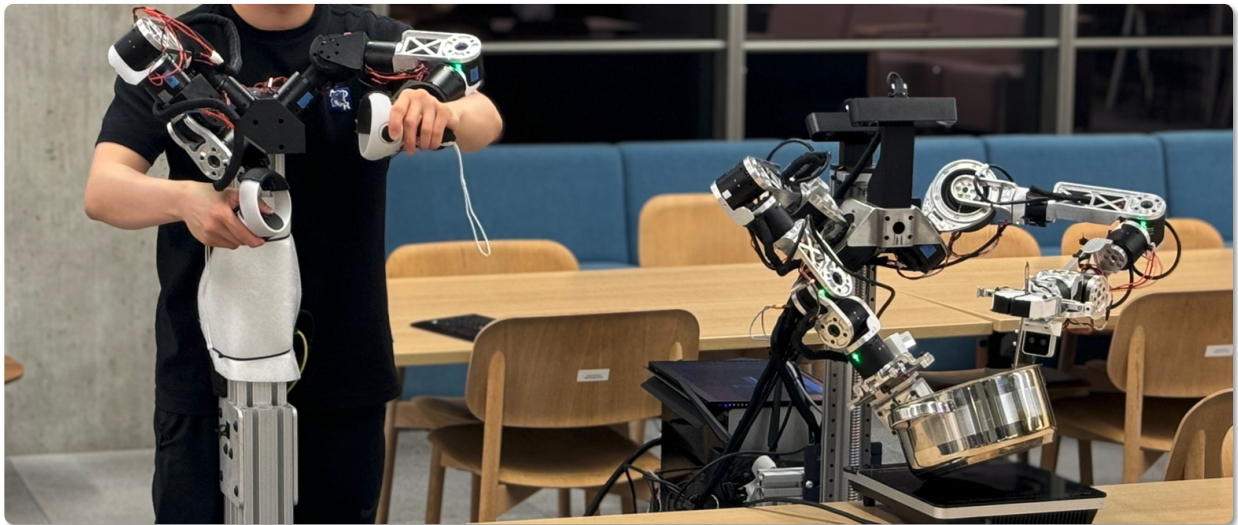


Figure 6.1: A depiction of our RoboCopilot System which consists of a 20 degrees of freedom mobile bimanual robot and a bilateral teleoperation device. Our system enables easy teleoperation as well as human take-over at any time, allowing for an effective human-in-the-loop teleoperation system for interactive learning.

Learning from human demonstration is an effective approach for learning complex manipulation skills. However, existing approaches heavily focus on learning from passive human demonstration data for its simplicity in data collection. Interactive human teaching has

appealing theoretical and practical properties, but they are not well supported by existing human-robot interfaces. This chapter proposes a novel system that enables seamless control switching between human and an autonomous policy for bi-manual manipulation tasks, enabling more efficient learning of new tasks. This is achieved through a compliant, bilateral teleoperation system. Through simulation and hardware experiments, we demonstrate the value of our system in an interactive human teaching for learning complex bi-manual manipulation skills.

6.1 Introduction

Humans play a critical role in teaching robot skills in robot manipulation. Recent developments in data-driven methods have shown promising results in learning complex robotic manipulation skills from human demonstration Zhao et al. (2023); Chi et al. (2023); Wu et al. (2023e); Brohan et al. (2022); Jang et al. (2022); Zhang et al. (2018). The advantages of learning-from-demonstration approaches arise partly from their simplicity: humans provide a dataset of demonstration trajectories and then a policy is trained on the collected dataset to mimic the human actions. With enough human demonstrated data the policy will eventually learn to perform the same task like the human demonstrator.

However, this passive imitation learning paradigm suffers from various inefficiencies that an interactive learning approach could overcome. In an interactive learning setting, humans take control of the robot when it fails and provide demonstrations for corrective behaviors. Prior works in Dataset Aggregation (DAgger) have shown, both theoretically and empirically, that corrective and interactive data improve policy learning performance by addressing the covariate shift issue Ross et al. (2011); Pomerleau (1988); Jang et al. (2021). However, existing human teleoperation systems for manipulation are not designed for human intervention. Remote controllers such as spacemice and VR devices map the relative pose change to the robot end-effector, which is not intuitive for users, especially in an interactive setting where the human may need to take over at any time Zhang et al. (2018); Wu et al. (2023e). Conversely, while exoskeleton-based devices have been effective in collecting passive human data Zhao et al. (2023); Wu et al. (2023e); Fang et al. (2023a), synchronizing the policy rollout to an exoskeleton can be potentially unsafe for the human operator in a DAgger setting Kelly et al. (2019). Despite the recent advances of the easy-to-use teleoperation tools for multi-joints robot systems Wu et al. (2023e); Zhao et al. (2023), the challenge of enabling intuitive interactive learning with these tools remains unresolved.

In this chapter, we introduce RoboCopilot, a robot system designed to accommodate **interactive human demonstrations** for learning bimanual robotic manipulation skills. We name our system RoboCopilot as it enables a human and a robot to cooperatively perform manipulation tasks, where the robot (the copilot) takes an assistive role before a lot of training. The system has two key components. First, we adopt the algorithmic framework introduced by Human-Gated DAgger (HG-DAgger) for interactive imitation learning and adapt it to our manipulation setting. In this pipeline, we alternate between model training

and data collection with a learned policy, where, during data collection, a human teleoperator interrupts and provides corrective feedback. This enables the robot to continually improve performance throughout the teaching process and collect higher-quality interactive data. We first verify the benefit of this procedure through controlled simulation experiments. Second, we build custom hardware which consists of a mobile, compliant bimanual manipulator and a bilateral teleoperation device which enables us to instantiate our interactive learning pipeline. Our system enables a teleoperator to **seamlessly take over robot control** from a policy during the interactive training process. We show that with RoboCopilot, we can successfully teach the policy to learn long horizon, contact-rich, bimanual mobile manipulation skills.

6.2 Related Work

6.2.1 Imitation Learning from Passive Human Demonstrations

Human demonstrations provide high-quality, action-labeled data for learning complex manipulation skills. Prior works show that behavior cloning is effective for learning from passive human data Zhao et al. (2023); Chi et al. (2023); Wu et al. (2023e); Brohan et al. (2022, 2023); Haldar et al. (2023), and increasing the number of trajectories consistently leads to better performance. However, it is challenging to provide performance guarantees for the trained policies. The number of demonstrations required is task-dependent, ranging from as few as ten demonstration trajectories for a simple picking skill Mandlekar et al. (2021) to up to a thousand trajectories Bousmalis et al. (2023). Furthermore, due to the issue of covariate shift, policies trained from passive offline data are unable to recover from the errors accumulated during online execution Ross et al. (2011); Spencer et al. (2021). Consequently, learning robust policies often has a significant hidden cost of an iterative process of data collection and policy learning. Our proposed system aims to enable interactive human teaching of complex manipulation skills, allowing the policy to continuously improve from newly collected demonstration data.

6.2.2 Interactive Imitation Learning

In an interactive imitation learning setting, the student policy receives feedback from the expert policy during policy execution, enabling the student to continually improve and learn to correct rollout mistakes Celemin et al. (2022). Interactive learning approaches such as Data Aggregation (DAgger) have shown better sample efficiency and robustness compared to behavior cloning methods, both theoretically and in practice Ross et al. (2011); Pomerleau (1988). However, retroactively relabeling demonstrations, as originally formulated, can be quite difficult for a human, especially for robot manipulation Laskey et al. (2016, 2017). Followup works have extended the DAgger framework to various settings to perform human-in-the-loop data collection Kelly et al. (2019); Hoque et al. (2021); Li et al. (2022); Mandlekar et al. (2020); Liu et al. (2023c); Spencer et al. (2020); Zhang & Cho (2016). We follow

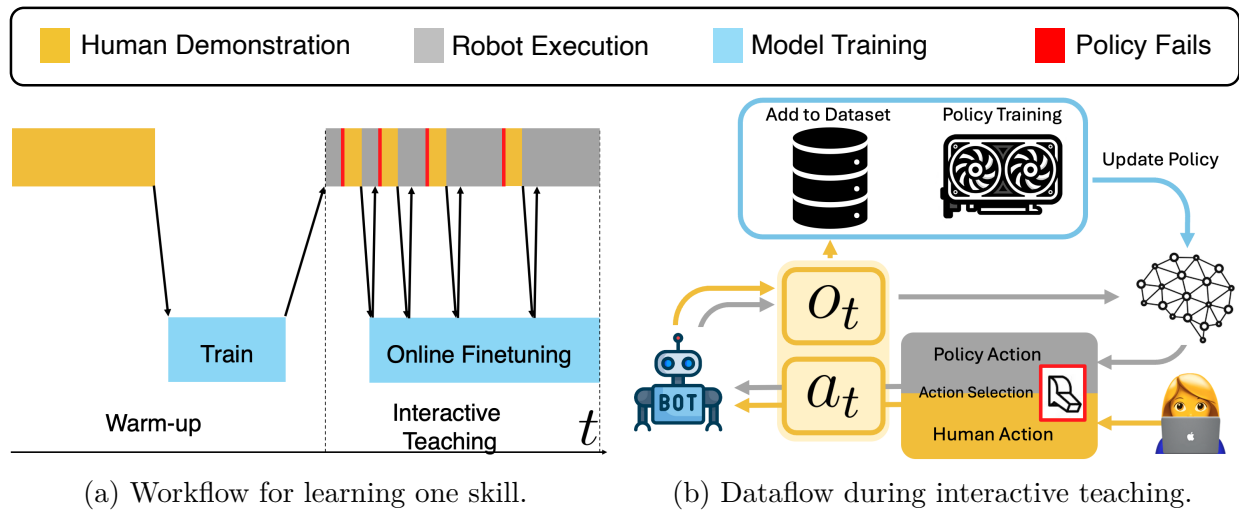


Figure 6.2: Overview for our interactive teaching system. (a) Workflow for learning a single skill: We start with a set of human demonstrations to pre-train the initial policy. Then in the interactive teaching stage, the policy is deployed and a human intervenes upon policy failure. The policy is continually fine-tuned from these new demos. As policy performance improves, less human intervention is needed. (b) During robot execution, the robot policy takes sensor observations and outputs actions. The human can decide when to use the policy switch to teleoperation. This enables the human to interrupt the robot on policy failure and correct the mistake, storing the data into the dataset. The model is continually training and being updated.

Human-Gated DAgger (HG-DAgger) Kelly et al. (2019) on its data collection phase, where the human operator determines when to intervene and take control of the student policy and when to release control back to the policy. However, such an approach requires the human and the agent to switch control when needed, which can be challenging for a typical teleoperation device. As such, the DAgger framework has not been widely adopted for complex bi-manual manipulation. RoboCopilot aims to bridge this gap through a teleoperation system that enables seamless human take over and continual policy improvement.

6.2.3 Robot Teleoperation Systems

Teleoperation systems have existed from the very beginning of robotics Lichardopol (2007) and the design for the teleoperation interface heavily impacts the final performance for modern data driven methods. Controllers such as joysticks and VR controllers Zhang et al. (2018); Zhu et al. (2022); Dass et al. (2024); Qin et al. (2023), while simple to set up, do not allow force feedback from the robot to the human operators and hide the kinematic joint limits of the robot from the human operators. Similarly, portable data collectors such as Universal Manipulation Interface (UMI) Song et al. (2020); Young et al. (2020); Chi et al.

(2024); Shafiullah et al. (2023) also pose a gap by removing the kinematic constraints from the data collection process. While exoskeleton-based puppeteering solutions have shown impressive results in collecting and learning fine-grained bimanual manipulation policies Zhao et al. (2023); Wu et al. (2023e); Fu et al. (2024); Fang et al. (2023a), the controllers lose synchronization during policy execution, making it difficult for human to intervene. Most related to ours, there has been a body of prior works on bilateral teleoperation Hulin et al. (2011); Katz (2018b); Schwarz et al. (2021); Elsner et al. (2022); Lenz & Behnke (2021); Whitney et al. (2014); Ishiguro et al. (2020); Tachi et al. (2003); Amini et al. (2017); Buamanee et al. (2024); Mohsen et al. (2023). In this work, we build a compliant, bimanual teleoperation system and demonstrate its capability for interactive human teaching.

6.3 Continual Interactive Imitation Learning

We aim to enable robots to continually learn from interactive demonstrations, where a human can take over control of the robot when the autonomous policy fails, as illustrated in Figure 6.2. As human demonstration, robot execution and skill learning occur in a tight learning loop, this allows the human operator to understand where and when the autonomous policy fails, providing more targeted demonstrations on the model failure cases. We will summarize our interactive learning algorithm in this section and explain how we achieve this through our proposed RoboCopilot system in the next section.

As shown in Algorithm 1, we first collect K human demonstration trajectories to warm up the policy. Following HG-Dagger Mandelkar et al. (2020), we allow the human expert to determine when to intervene and take control of the robot execution. The intervention data from the human experts are added to the dataset. After each round of data collection, we fine-tune our policy using all the collected data by performing several gradient-based updates to the policy network. Our policy network is parameterized by a diffusion network Chi et al. (2023). As the policy improves, the number of required interventions decreases significantly.

Prior work has shown that continually fine-tuning neural networks on data from non-stationary distributions causes catastrophic forgetting Goodfellow et al. (2013); Kirkpatrick et al. (2017). Despite this, we adopt fine-tuning during the interactive teaching process give

Algorithm 1 Human-in-the-loop Imitation Learning

- 1: **Input:** Init policy π_0 , human expert π^* , # of iterations N , warmup human demonstration dataset D
 - 2: Train warmup policy π_1 on D with behavior cloning
 - 3: **for** $i = 2$ to N **do**
 - 4: **for** each step in the environment **do**
 - 5: **if** human expert π^* intervenes **then**
 - 6: Execute expert human action: $\pi^*(s)$
 - 7: Add the intervention data: $D \leftarrow D \cup \{(s, \pi^*(s))\}$
 - 8: **else**
 - 9: Execute robot policy $\pi_i(s)$
 - 10: **end if**
 - 11: **end for**
 - 12: $\pi_{i+1}(s) \leftarrow \text{finetune}(\pi_i, D)$
 - 13: **end for**
 - 14: **Output:** Final policy π_N
-

its multiple advantages over training policies from scratch after the data collection is done: First, collecting online samples from the current policy alleviates covariate shift, as the human can intervene when necessary to teach correction skills. Second, as the policy is continually fine-tuned and executed, the human operator can quickly observe current failure modes and collect more targeted demonstrations. Finally, online learning reduces the overall training time.

Practically, once the interactive teaching finishes, we can utilize the online interactive data and retraining a policy from scratch. Doing so avoids the non-stationarity of the training distribution but still takes advantage of the interactive data. In practice, we find this approach to provide the best policy that can be used for deployment. This amounts to restarting Algorithm 1 with the most up to date dataset D .

6.4 Teleoperation System

We develop a versatile mobile bi-manual robot with a capable teleoperation system that allows us to instantiate human-in-the-loop interactive teaching for a wide range of real world tasks. Our physical RoboCopilot teleoperation system consists of the physical robot, the teleoperation device, and the teleoperation workflow. More details about our custom low cost robot hardware can be found in Appendix D.3.

Our teleoperation system is an effective way to enable human-in-the-loop imitation learning for manipulation. A more capable and human intuitive device enables a user to

more precisely control the robot which offers increased data quality for downstream learning. While for dynamical systems with a lower dimensional action space like UAVs or cars, one can more easily apply DAgger algorithms by having a human relabel the data with a joystick or wheel Ross et al. (2012); Kelly et al. (2019). In general, there is no obvious way to relabel trajectories for manipulation trajectory data. This is due in part to the high dimensional action space and complex environment dynamics.

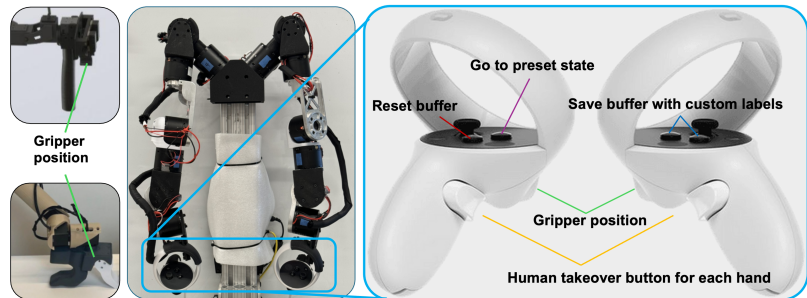


Figure 6.3: The comparison of the different end-effector human interface designs for different teleoperation systems is shown below: Left: Aloha Zhao et al. (2023) and GELLO Wu et al. (2023e)’s handheld interface. Middle: RoboCopilot layout, where we attached the Quest2 controller at the end of our GELLO device. Right: The key map of our end-effector human input interface. We optimized the layout to allow efficient gripper control and interactive human-in-the-loop teaching.

6.4.1 Teleoperation Device

The core principle of the teleoperation leader device is to enable maximal data collection quality and ease of use for the human during operation and take over. Our puppeteering teleoperation device is a low cost approximate kinematic replica of our target arm, following GELLO Wu et al. (2023e). More detailed hardware specifications are found in Appendix D.3. We attach a Meta Quest 2 controller Meta (2024) at the end of our teleoperation leader arms, serving as both an ergonomic gripping handle and a multi-input control device. Previous works have primarily focused on the active portion of controlling the arm, but our approach also considers the importance of a comprehensive and user-friendly interface. A comparison of different teleoperation devices is shown in Figure 6.3. While teaching devices like ours are not new, the focus has largely been in an imitation learning context. Our goal is to show how to effectively extend the usefulness of these devices in an interactive learning setting.

6.4.2 Active Control

We actively control the motors of the leader arm enabling less user fatigue and smoother human take over. The weight and inertia of the teleoperation device requires additional energy input from the user to drive the device. Active gravity compensation using the manipulator equation alleviates this burden, allowing the operator to maneuver the system without excessive effort, reducing operator fatigue Siciliano & Khatib (2007); Lynch & Park (2017).

Active control also enables a bilateral system, where the leader arm can provide force feedback to the user felt by the follower arm. This enables a user to understand and feel the forces that the robot feels, allowing the user to more effectively operate the robot. We modify the PD control law used in Katz (2018b) control law so the user feels a scaled down version of the forces Katz (2018b).

$$\tau_L = \alpha K_p(\theta_F - \theta_L) + \beta K_d(\dot{\theta}_F - \dot{\theta}_L) \quad (6.1)$$

$$\tau_F = K_p(\theta_L - \theta_F) + K_d(\dot{\theta}_L - \dot{\theta}_F) \quad (6.2)$$

where L is the leader teleoperation device and F is the follower, τ is actuator torques, θ is joint position and $\dot{\theta}$ is joint velocity. We add α and β constants so that the teleoperation device does not feel the full inertia of the arm, while still enabling effective control. Additionally, the use of active motors on our teleoperation device allows us to flexibly vary the α, β bilateral coefficient in different situations.

6.4.3 Human-in-the-loop Interactive Learning

Our RoboCopilot system is designed to be user-friendly and fully equipped with features that facilitate both the smooth operation of the robot for task completion and efficient data collection. The teleoperation device includes multiple buttons dedicated to data collection

utilities, such as saving data and resetting the robot, streamlining the process of capturing valuable data during operation. This enables the teleoperator to be explicit about the data being collected, and can easily control for the data quality.

Control of the leader arm is intuitive, with bilateral control allowing the user to feel the forces experienced by the robot arm. This feature is particularly beneficial when the robot is in contact with the environment or handling heavy objects. When the autonomous policy is executing, the leader-follower relationship is reversed where the teleoperation device matches the real robot. The teleoperation can then activate take over at any point in time by engaging with the teleoperation device, without interrupting process flow as the devices are already synchronized. During robot policy execution, we use stiff gains on the leader to minimize the state difference, whereas during teleoperation mode, we use lower gains to ease the forces on the human teleoperator.

6.5 Experiments

We conducted a series of experiments to demonstrate the value of our system for continual, interactive imitation learning in both simulation and the real world. Our system enables interactive data collection with a human in the loop. Our primary research question is how this interactive data collection affects the collected data quality. We measure the quality of the collected data by evaluating the final policy performance. Specifically, we compare the policy performance using three data collection and policy learning methods: (1) Offline Behavior Cloning (BC), (2) Human-in-the-loop interactive learning where the policy is continually trained and used for collecting the next batch of data (Continual DAgger), and (3) Training a policy from scratch using all data, including the data collected by Continual DAgger (Batched Dagger). Through these experiments, we aim to demonstrate that interactive data collection can accelerate learning, reduce the need for laborious data collection process, and improve the overall performance. Additionally we demonstrate the effectiveness of our real world system at learning various tasks.

6.5.1 Simulation

We conduct imitation learning experiments on the standard Robomimic benchmark Mandlekar et al. (2021). The benchmark provides human demonstration trajectories and simulation environments for a set of robotic manipulation tasks including bimanual tasks, showing the benefit of our approach for manipulation in general. The goal of these experiments is to validate the effectiveness of the algorithmic approach in a controlled reproducible environment. We choose the Can, Square, and Transport tasks from the benchmark which are visualized in Appendix D.2. Experiments involving human interaction can be difficult to reproduce due to the volatile nature of human interaction. To improve reproducibility, we use the pre-trained expert diffusion policy π^* from Chi et al. (2023) as a human substitute. As this trained expert

Table 6.1: Simulation results comparing behavior cloning and various DAgger methods on the robomimic benchmark Mandlkar et al. (2021). We collect 10 trajectories for warming up the dataset and training the initial policy. BC-H refers to behavior cloning from a human demonstration dataset. BC-P refers to behavior cloning from an expert dataset collected by our expert policy. The expert policy is the same policy that makes interventions in our DAgger methods. **Continual DAgger** denotes our continual DAgger method that continually finetuning with interaction data. **Batched DAgger** denotes training from scratch with the Continual DAgger collected data. We see leveraging human in the loop during the data collection process significantly improves over the baselines. For our DAgger runs, we initialize with 10 expert demo trajectories. Experiments are run across three seeds.

Environment	Total Number of Trajectories	BC-H	BC-P	Continual DAgger (Ours)	Batched DAgger (Ours)
Can	10 (warmup)	0.28±0.04	0.43±0.0	—	—
Can	15	0.45±0.02	0.57±0.04	0.57±0.02	0.63±0.02
Can	20	0.61±0.04	0.62±0.02	0.62±0.02	0.72±0.02
Can	30	0.76±0.01	0.82±0.02	0.85±0.04	0.83±0.02
Can	40	0.83±0.03	0.93±0.01	0.87±0.02	0.93±0.02
Square	10 (warmup)	0.26±0.02	0.30±0.02	—	—
Square	15	0.38±0.05	0.28±0.03	0.39±0.02	0.43±0.02
Square	20	0.40±0.00	0.41±0.03	0.43±0.01	0.44±0.03
Square	30	0.48±0.05	0.46±0.03	0.47±0.02	0.55±0.02
Square	40	0.52±0.01	0.51±0.02	0.53±0.05	0.57±0.01
Transport	10 (warmup)	0.41±0.10	0.65±0.04	—	—
Transport	15	0.55±0.02	0.57±0.05	0.72±0.06	0.87±0.02
Transport	20	0.53±0.05	0.70±0.02	0.88±0.02	0.97±0.02
Transport	30	0.65±0.04	0.61±0.08	0.90±0.04	0.92±0.02
Transport	40	0.77±0.04	0.83±0.01	0.90±0.04	0.93±0.02

cannot decide when to intervene, we follow Algorithm 1 but use $\pi = (1 - \beta)\pi_i + \beta\pi^*$ as our rollout policy when $|\pi_i(s_t) - \pi^*(s_t)| > \epsilon$. Here, π_i is our training policy at the i^{th} iteration.

The results are shown in Table 6.1. Our DAgger variants use 10 expert demos to initialize, with the rest being DAgger-corrected trajectories. Our continual learning method achieves superior performance to behavior cloning baselines, despite the issue of catastrophic forgetting. Some benefits of continual learning are hard to show with a learned policy trained on the distribution of human demos, as the policy does not have the ability to give sophisticated corrective demonstration nor intelligently collect targeted demonstrations in areas of failure. Even so, our simulation results demonstrate the value of interactive data, as seen by the performance boost provided by Continual DAgger. This increase in data quality is further supported by the improvement of the policy in our Batched DAgger training where we train the policy from scratch using the Continual DAgger data. As expected, Batched DAgger greatly outperforms the continually trained variant in almost all cases. This greatly motivates the advantages of DAgger like training for manipulation. Next we test the effectiveness of our RoboCopilot system in real world settings using a human as the expert demonstrator,

Table 6.2: Task completion rate of the industrial picking experiment we ran in the real environment. From this table, we can see that under the human-in-the-loop interactive learning setting, the policy can achieve higher performance given the same amount of human-labeled trajectories. With the human-in-the-loop copilot data collection process, the learned policy achieves a much higher success rate given the same number of training trajectories. Additionally, since the human operator can allow the policy to take over again once a mistake is corrected, the number of human-teleoperated timesteps in these DAggered trajectories is significantly less compared to those in offline BC data.

Task	# of Trajectories	Offline BC	Continual DAgger (Ours)	Batched DAgger (Ours)
Industrial Picking	12 (warmup)	38.9%	—	—
	24	61.1%	72.2%	72.2%
	36	66.7%	72.2%	77.7%
Industrial Part Transport	10 (warmup)	33.3%	—	—
	15	50.0%	66.7%	75.0%
	20	58.3%	75.0%	91.7%

leveraging our teleoperation device to interrupt and interact with the scene.

6.5.2 Real World

Our real-world experiments provide similar conclusions to our simulation experiments, where interactive data collection provides higher quality data than passive data collection from the expert policy. We consistently observe that the success rate keeps increasing as we collect more interactive data, and the need for human intervention decreases accordingly. The need to intervene also serves as a useful signal in practice for when to stop collecting new data.

Part Picking

We test our RoboCopilot system in the real world on an industrial part picking task. The goal of the robot in this task is to pick up industrial aluminum extrusions and place them into a bin. We use two different sized aluminum extrusions, a small one which can be picked up with a single arm, a longer beam should be picked up by both arms. To control the experiment we define a domain of interest where the 8020 beams are placed. Additionally, for the placing bin, we mark 3 train locations, as well as 2 test locations. During testing of a policy, we choose and evaluate the robot of 18 predetermined locations. This task allows us to study various components of our system. First, the setting is very multimodal, as there may be different strategies and methods of how to pick up the object. Second we can test the effectiveness compliance of our system as holding the long beam with both hands is more straightforward for a compliant system.

We define four “task types”: large bar, small bar left hand, small bar right hand, and small bar middle both hands. The primary question we aim to address is how effective our

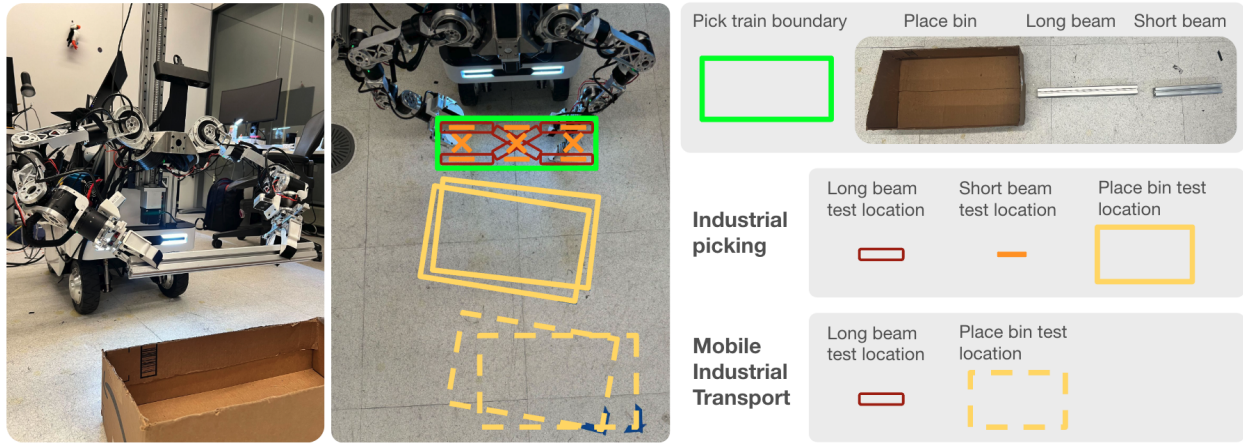


Figure 6.4: An illustration of the evaluation protocol for the industrial part transport tasks. During training, the beam is placed in different positions within a defined boundary (highlighted in green). Industrial picking requires the robot to locate and manipulate the long beam or short beam and place it within the bin. Mobile industrial picking only considers the long beam, but the bin is further away, requiring the robot to drive the base before placing. We label the poses of the beams and the bin to ensure consistency during evaluation.

shared autonomy system is for collecting demonstrations to learn new skills. To start the experiment, we collect 12 trajectories (3 for each subtask) to initialize the policy, ensuring that it does not behave randomly or completely ineffectively. During the policy rollout, the human teleoperator has the option to take over from the autonomous system if the policy fails to complete the task. For each human-in-the-loop DAgger process, we will continue running the autonomous system until the human operator takes over, three times for each subtask. We tested our continuous learning pipeline under two different settings: Batched DAgger and Continual DAgger. For the Batched DAgger setting, every time we collect new interactive data, we relaunch the training process from scratch. In contrast, for the Continual DAgger setting, we continuously stream the human interactive data into the training pipeline to continuously fine-tune the model. All DAgger runs use policies from the warmup offline BC checkpoint.

Part Transport

In addition to the stationary industrial part pick-and-place task, we applied the RoboCopilot pipeline to a more challenging industrial part transport task, which requires moving the robot base while performing manipulation tasks, as shown in Figure D.1.1. Specifically for this task, the robot must navigate to the tote, which is placed far from its starting position, to successfully complete the placing task. See Appendix D.1 for a visual depiction the task. In Table 6.2, we compare the performances of three methods given the same demonstration

budget: Offline BC, which trains a policy from passively collected data; Continual DAgger, which continually fine-tunes a policy after collecting 10, 15, and 20 trajectories; and Batched DAgger, which uses the final dataset obtained by running Continual DAgger and trains a policy from scratch. During Continual DAgger, we observe that humans can identify initial configurations where the current policy fails and reset intelligently. In this way, we find that Continual DAgger is more data-efficient than Offline BC, thanks to intelligent reset. However, Continual DAgger keeps fine-tuning the same policy as new demonstrations are added to the dataset, leading to suboptimal policy optimization. In Batched DAgger, we train one policy from scratch using this interactively collected dataset and achieve the best performance, further validating our claim that human-in-the-loop data collection improves the quality of the collected data.

Kitchen

Lastly we study a long horizon task in a toy kitchen, requiring the robot to execute multiple individual steps in order to “cook the tomato”. This high level task is broken up into 4 smaller subtasks, where the robot first must open the cabinet door, then pick the tomato, then put the tomato in the pot, and lastly turn the dial. If the robot fails to accomplish any one task, we count failure for all the following steps as in the CALVIN long horizon benchmark (Mees et al., 2022). See Fig 6.5 for an illustration of this task. We compare



Figure 6.5: An overall illustration of the toy kitchen task. (Subtask 1) The robot needs to first open the spring-loaded cabinet door and hold the door open. (Subtask 2) The robot can pick the tomato and transfer it to the stove area. (Subtask 3): At this stage, the robot needs to put the tomato into the pot. Notice that the pot can be at different locations. (Subtask 4) Finally, the robot needs to turn the correct dial depending on which stove the pot is at.

Table 6.3: Experimental results on the long horizon kitchen task. We compare Offline BC with Batched DAgger for different trajectory dataset sizes, in increments of 15. The size of the training dataset is shown. For each experiment, we also show the number steps that required a human to operate the robot to collect the data at each stage. We also show a detailed breakdown of the task success rate for each subtask in the overall task.

# of Trajectories	Method	Dataset Size (# Transitions)	% intervention steps	Open Door	Grab Tomato	Place Tomato	Turn Dial
15	Offline BC	8491	100%	100%	70%	25%	5%
30	Offline BC	16654	100%	100%	70%	35%	5%
	Batched DAgger	15162	65 %	100%	85%	45%	20%
45	Offline BC	25694	100%	100%	85%	40%	0%
	Batched DAgger	20844	52%	100%	90%	50%	30%

naive Offline BC to our interactive approach leveraging Batched DAgger. The results are found in Table 6.3. We find that leveraging human-in-the-loop not only results in better long horizon performance, but it also decreases the necessary human teleoperated time to reach that performance. Interestingly we find that more demos for Offline BC does not help the policy learn to fully complete the task, whereas our Batched DAgger approach enables a significant success increase in fully completing the sequence of subtasks, despite having a smaller overall dataset size. Our experiments show that Batched DAgger with one iteration of data collection and a total of 30 trajectories is able to match or outperform the success rates of Offline BC with 45 trajectories on all subtasks, with the performance gap being the largest for completing the full task (i.e. Turn Dial). This shows that our approach is also effective in learning trajectories that have long time horizons. Lastly we see that Batched DAgger requires less human collected trajectory steps for a fixed number of training trajectories, as the human only needs to operate after a policy failure. This reduction in human intervention steps further underscores the efficiency of our human-in-the-loop system in teaching the robot

6.6 Discussion

Limitations. While our RoboCopilot system shows promising results in enabling interactive teaching and continual learning for bimanual robotic manipulation, it is not without limitations. First, the requirement for human intervention during the learning process can be resource-intensive, as it necessitates a skilled operator to oversee and correct the robot’s actions continually. This dependency might limit the scalability of our system in environments where such expertise is not readily available or the task is too difficult to even teleoperate. Second, although our system aims to be cost-efficient, the initial setup and maintenance costs might still be prohibitive for smaller organizations or research labs due to the large number of custom components. Additionally, the use of planetary gearboxes, while beneficial for

compliance and cost, introduces some degree of backlash, which may affect the precision of fine manipulation tasks.

Conclusions In this work, we introduced RoboCopilot, a novel system that enables human-in-the-loop interactive imitation learning for bimanual robotic manipulation tasks. The RoboCopilot system underscores the potential of seamlessly interfacing a human operator and an autonomous policy for interactive teaching. Future work will focus on enhancing its scalability and ease of deployment in diverse real-world environments.

Chapter 7

Task Planning with Language Models

This chapter is based on the paper “Interactive Task Planning with Language Models” (Wu et al., 2023c), by Philipp Wu, Boyi Li, Pieter Abbeel, Jitendra Malik.

An interactive robot framework accomplishes long-horizon task planning and can easily generalize to new goals or distinct tasks, even during execution. However, most traditional methods require predefined module design, which makes it hard to generalize to different goals. Recent large language model based approaches can allow for more open-ended planning but often require heavy prompt engineering or domain specific pretrained models. To tackle this, we propose a simple framework that achieves interactive task planning with language models by incorporating both high-level planning and low-level skill execution through function calling, leveraging pretrained vision models to ground the scene in language. We verify the robustness of our system on the real world task of making milk tea drinks. Our system is able to generate novel high-level instructions for unseen objectives and successfully accomplishes user tasks. Furthermore, when the user sends a new request, our system is able to replan accordingly with precision based on the new request, task guidelines and previously executed steps. Our approach is easy to adapt to different tasks by merely substituting the task guidelines, without the need for additional complex prompt engineering.

7.1 Introduction

The rise of Large Language Models (LLMs) and proliferation of chatbots highlight the importance of human interaction in AI systems. Beyond merely executing user commands, an autonomous agent should fluidly receive and incorporate feedback at any step during the execution process. Consider the seemingly straightforward human task of preparing a flavorful milk tea, which we study in this work. Such a task, while simple to humans, requires a robot to decompose it into numerous intermediate steps. Not only does the robot need to generate and precisely execute the steps, but the robot should also remain receptive to real-time modifications or feedback to the initial request. For example, the user might request

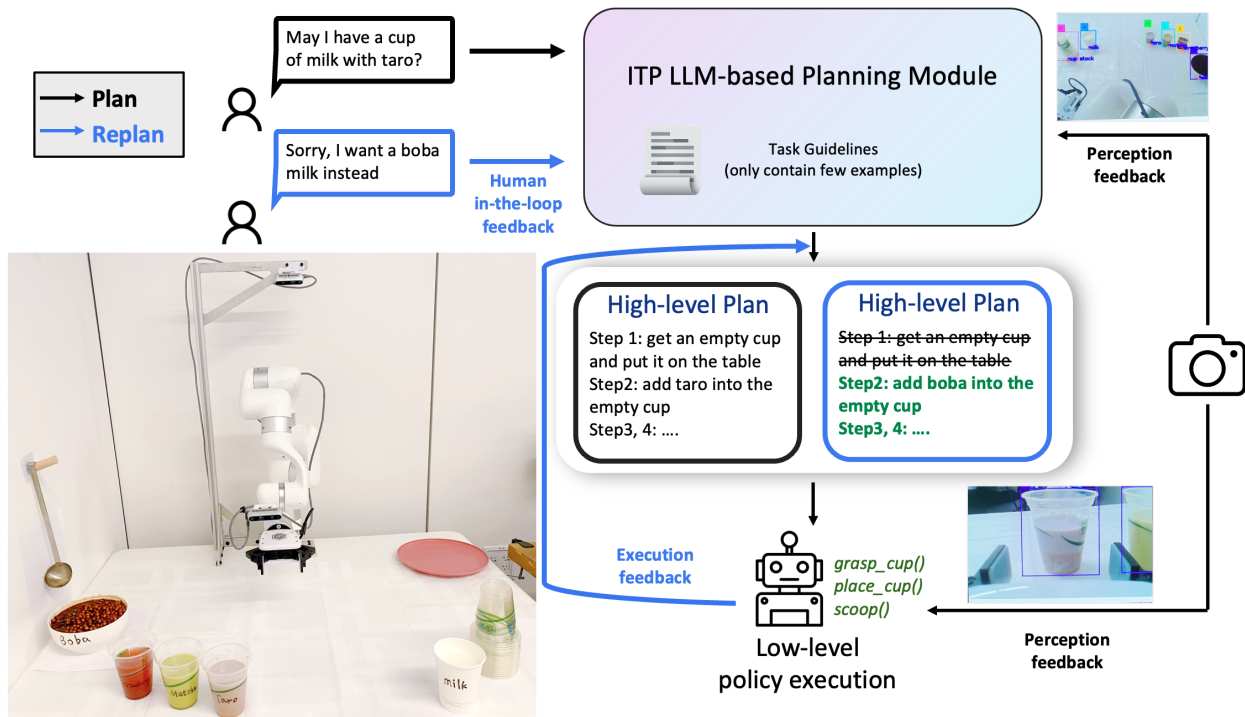


Figure 7.1: An overview of ITP. ITP generates high-level plans and executes the low-level robot skills through LLMs. Our system generates a high-level plan based on user requests and task guidelines. When the system is interrupted with a new request, the system will replan, taking into consideration prior completed steps and task guidelines. Each step of the plan is executed by leveraging an LLM executor, equipped with additional visual grounded outputs, to call lower-level skills. In the example shown, the user first requests ‘*May I have a cup of milk with taro?*’, a request for which the high-level plan is not provided in the task guidelines. After the robot has finished the first step, the user wants to revise the order to a boba milk. Our system is able to replan and make a new set of high-level steps based on the new request, a history of completed steps, and task guidelines, which can then be completed by the lower level execution module to successfully.

some boba to be added to their drink. A robot should be able to seamlessly incorporate such interaction during operation.

In light of these challenges, we propose a simple framework for **Interactive Task Planning** with language models, denoted as ITP. Our framework leverages LLMs to plan, execute, and adapt to user inputs throughout the task lifecycle. Figure 7.1 illustrates an exemplary interaction with our system. Our primary objective is to offer a blueprint for deploying real-world robotic systems that harness pretrain language models to coordinate the execution of lower-level skills of a robot in a simple manner. In this work, we utilize GPT-4 (OpenAI, 2023) as the language model backbone. ITP consists of two primary modules; (1) a high

level planner, which takes as input a prompt to specify and a user request to output a step by step plan, and (2) a low level executor which tries to achieve a given step by converting robots skills into a functional API, which enables GPT-4s function calling capabilities to directly interact with the robot, abstracting code level details from the system. ITP does not require the training of additional value functions such as SayCan (Zeng et al., 2022; Huang et al., 2023), and does not require code level prompts such as Code as Policies (Singh et al., 2023) or ProgPrompt (Liang et al., 2022). Furthermore, ITP dynamically generates novel plans and re-adjusts its plan based on user input. We hope our framework will be useful for accomplishing a wide range of interactive robot tasks and will release our codebase to foster advancements in this field. We outline the key features of ITP below:

1. ITP is a training-free robotic system for interactive task planning with language models with a focus on simplicity. We showcase ITP in the context of a real-world boba drink-making robot that integrates planning, vision and skill execution.
2. ITP leverages a simple prompt format, which we show is effective across simulated and real settings. Additionally ITP system converts the lower-level skills into a functional language-based API that can be leveraged by any function calling LLM (ie. GPT-4). This enables a user to prompt the system through natural language rather than code, removing the need for code-level prompt engineering.
3. Our system exhibits robustness in adapting to user requests during execution, allowing it to consider the updated goals, previously completed steps, and task guidelines in order to replan new steps.

7.2 Related Work

7.2.1 Task planning

Task planning, the problem of developing a plan to achieve a desired goal, is an integral component of our work. Traditionally, task planning in robotics commonly leverages symbolic planners which reduce the planning problem into a search problem (Ghallab et al., 2004; Bonet & Geffner, 2001). Practitioners often define the problem in a declarative language (Jiang et al., 2019; Ghallab et al., 1998; Lifschitz, 2008; Fikes & Nilsson, 1971), which can be restrictive as it requires meticulous definitions of the problem parameters, such as actions, their preconditions and their effects. Task and motion planning (TAMP), takes task planning a step further and also jointly considers the lower level execution during higher level planning (Garrett et al., 2020a; Mansouri et al., 2021). TAMP methods also consider symbolic representations and leverage search algorithms to extract the final sequence of lower-level primitives and has seen success in robotic manipulation (Siméon et al., 2004; Garrett et al., 2017, 2020b). As the search space can often be prohibitively large, some methods leverage hierarchy and/or sampling (Bacchus & Yang, 1991; Plaku & Hager, 2010; Kaelbling & Lozano-Pérez, 2011;

Kaelbling & Lozano-Pérez, 2013). Our approach replaces traditional planning pipelines with LLMs, offering common-sense reasoning, enhanced interaction capabilities, and the ability to define the problem’s scope using natural language.

7.2.2 Language Models as Planners in Robotics

Due to the popularity of LLMs, there has been a rising interest in leveraging LLMs as a policy in robot systems. One work in this direction leverages LLMs as zero-shot planners in simulated embodied settings (Huang et al., 2022) by converting the scene and task definitions into language, then letting the LLM directly predict actions. Works such as Socratic models, SayCan, and Grounded decoding (Zeng et al., 2022; Ahn et al., 2022; Huang et al., 2023) follow in this line of work, coordinating many large pre-trained models with a robot to solve various tasks. In contrast to approaches like SayCan (Zeng et al., 2022), which necessitate a pretrained value function to ground actions, we rely on prompting the language model with task guidelines and robot skills. This implicitly encodes preconditions and effects reminiscent of traditional declarative task planning approaches but can be done so with natural language, which is more expressive and easier for the average user to tune. G-PlanET (Lin et al., 2023) explores using language models for robot task planning by generating high-level subgoals in simulated environments. Tidy bot (Wu et al., 2023a) shows that LLMs can help a robot follow a user’s preferences based on a few examples. We also prompt the model with a small set of examples but explore generalization to new goals. Reflect (Liu et al., 2023e) uses large models to make an agent recount their experiences and correct failures. LLMs have also been used to allow robots to seek help when uncertain (Ren et al., 2023).

A related approach, used in Code as Policies (Liang et al., 2022) and ProgPrompt (Singh et al., 2023) leverages the code writing capabilities of LLMs to generate code that a robot agent can execute directly. This often requires heavy prompt engineering of example code to show the model how to properly use the provided functions to accomplish a directive. Language-guided Robot Skill Learning (Ha et al., 2023), like us, takes a hierarchical approach to LLM planning, but assumes access to the simulator which provides ground truth state information. Voyager (Wang et al., 2023b) uses LLMs to build a lifelong learning agent for Minecraft by having the agent explore and solve new tasks through writing code that interacts with the API.

Our work falls into this general category of leveraging LLMs to plan, and then execute actions in the environment. In contrast to prior work, we allow the LLM to generate a high-level plan based on contextual information. These low-level plans are then executed directly by an LLM with access to the functional API of the robot using a pre-trained VLM to ground the visual scene into primitives. Our work focuses on how to instantiate such a system in the real world.

7.3 Method

ITP offers a blend of high-level planning and low-level execution, powered by LLMs. In contrast to prior work (Liang et al., 2022; Singh et al., 2023), our approach enables the LLM to create a high-level plan informed by contextual information in the form of a list of steps. Each step of this plan is subsequently realized by another LLM with access to the functional API of the robot. A pre-trained VLM grounds the visual scene into language. Our work focuses on how to instantiate such a system in the real world. Our framework, shown in Figure 7.1, with a more detailed breakdown in Figure 7.3, consists of a hierarchy of two levels, the high level and low level.

7.3.1 High-level Planning

LLMs for Planning. We utilize GPT-4 (OpenAI, 2023) as our language model, one of the most capable LLMs available at the time of this writing. The high-level planner takes as input a given prompt, task guidelines, and a user request, and outputs a step-by-step plan to execute the request. It also retains past user interactions for any necessary replanning.

Simple Prompting Strategy Task guidelines, described using natural language, outline the scope of the robot’s tasks and are provided to the high-level planner. The prompt contains the user’s request and task guidelines which contain few-shot prompt examples of plans in the given domain of interest, and a description of the available materials to the robot. In our milk tea system, the task guidelines consist of a select set of menu items, their corresponding preparation steps, and a list of relevant ingredients. This includes the procedures for a few drinks like ‘pure milk’ and ‘boba milk’. See Task Guidelines 7.1 for the exact task guidelines we used in our experiments. Our system utilizes these guidelines to determine the feasibility of making a new drink based on available materials. Leveraging LLMs’ few-shot learning capabilities (Brown et al., 2020), ITP can generalize from the baseline guidelines to make detailed steps for other drinks such as ‘boba strawberry milk’ or ‘taro milk’.

7.3.2 Low-level Execution

The low-level executor takes each generated step and does its best to complete it successfully, conditioned on additional information about the scene and available robot skills. We use pretrained vision modules to convert the scene into a language compatible format. Additionally, we translate robot skills into a functional API, automatically translating the python docstring of the robot skills into callable functions by the language model.

Visual Scene Grounding. The role of the vision module in our system is to process the camera inputs into concise language descriptions of the scene, which can further be processed for planning and task execution downstream. In our drink-making system, the visual grounding system accepts a list of menu items and generates corresponding bounding boxes. Using a simple projective mapping, we then approximate the x and y locations of each item in the robot frame. We employ the pretrained VLM: Grounded-DINO (Liu et al.,

```

Options:
Pure milk, Strawberry milk, Boba milk
Instructions:
Pure milk
Material: milk
Steps:
0) get an empty cup and bring it to the working area
1) pour the milk into the working cup
2) put the working cup in the finished location
Strawberry milk
Material: strawberry jam, milk
Steps:
0) get an empty cup and bring it to the working area
1) add strawberry jam to the working cup
2) pour the milk into the working cup
3) put the working cup in the finished location
Boba milk
Material: boba, milk
Steps:
0) get an empty cup and bring it to the working area
1) add boba to the working cup
2) pour the milk into the working cup
3) put the working cup in the finished location
Available material we have now:
boba, strawberry jam, mango jam, matcha powder, taro, milk,blueberry

```

Task Guidelines 7.1: The task guidelines we use for our drink making experiments. Task guidelines only need contain simple; human interpretable few shot examples and a description of relevant assets in the scene.

2023d), a variant of the original DINO model (Caron et al., 2021) fine-tuned for extracting 2D bounding boxes given language descriptions. The final text description is represented as a dictionary of object description to (x, y) location. The vision system gives a holistic ‘understanding’ of the scene, despite the location assignments being imprecise.

Robot Skill Grounding. The language model interfaces with a predefined skill set in Python that controls the robot. These skills are translated into a functional API by parsing of function definitions and related doc strings. This can be directly used with GPT’s function-calling layer (OpenAI, 2023). In contrast to methods like ProgPrompt or Code as Policies, our system does not require examples or function internals when prompting the LLM. Instead, more detailed prompting of the language model can be specified via natural language in the documentation of the functions.

7.3.3 Replanning

Beyond the aforementioned components, ITP considers new requests from the user as *human-in-the-loop feedback*. We allow a human to interpret the robot execution at any stage with a new prompt. This then triggers our replanning pipeline. The system will consider completed steps, task guidelines, the new request, and the chat history to generate a new plan. The

details of replanning are see in Figure 7.3. We also showcase ITP’s adeptness in planning and adaptive replanning of the same example in Figure 7.2.

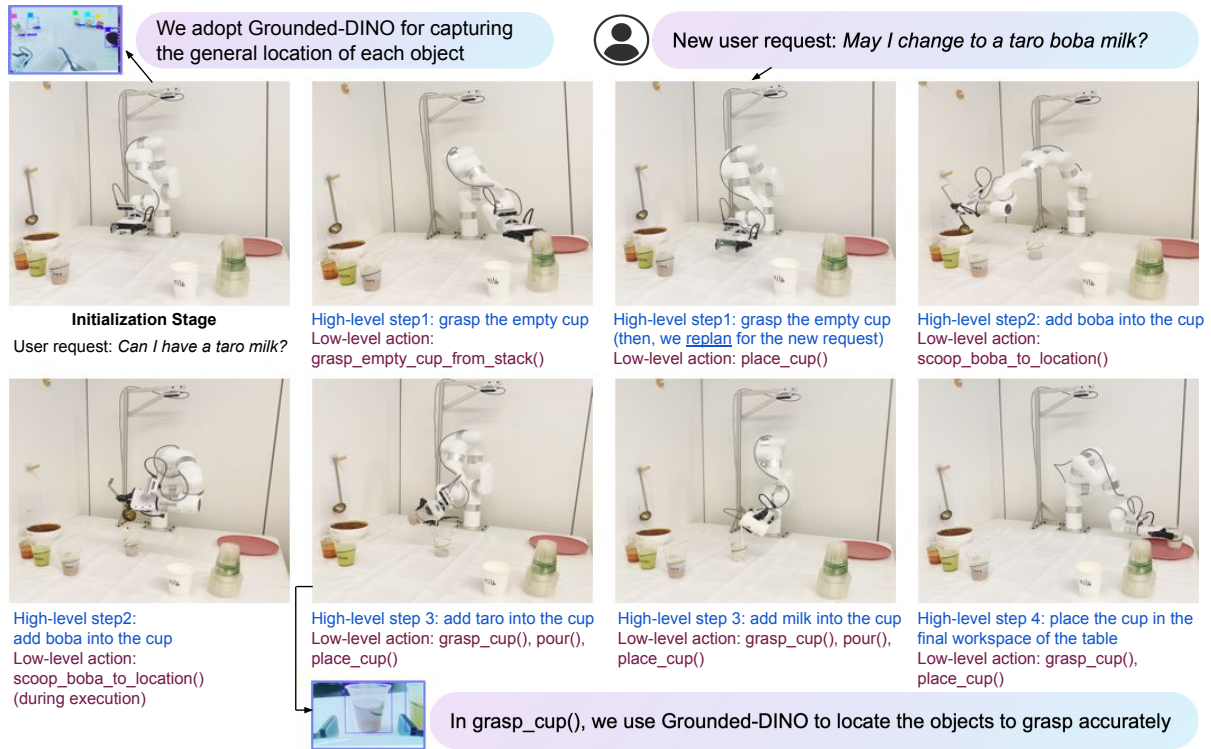


Figure 7.2: An example of ITP to make a cup of taro milk with boba. Our system first makes a high-level plan based on the user request using GPT-4: step 1) grasp the empty cup, step 2) add taro into the cup, step 3) add milk into the cup, step 4) place the cup in the final workspace. For each step in the high-level plan, we feed step into another instance of GPT-4 and obtain the corresponding low-level actions which is directly executed on the robot. As for the perception component, ITP uses Grounded-DINO to capture the general location of each object and locate the object accurately when taking the actions. However, after grasping the empty cup, the user sends a new request ‘*May I change to a taro boba milk?*’. Considering the history of completed steps, the system replans and generates the following high-level steps and low-level executions. The following plan has been changed to: step 2) add boba into the cup, step 3) add taro into the cup, step 4) add milk into the cup, step 5) place the cup in the final workspace.

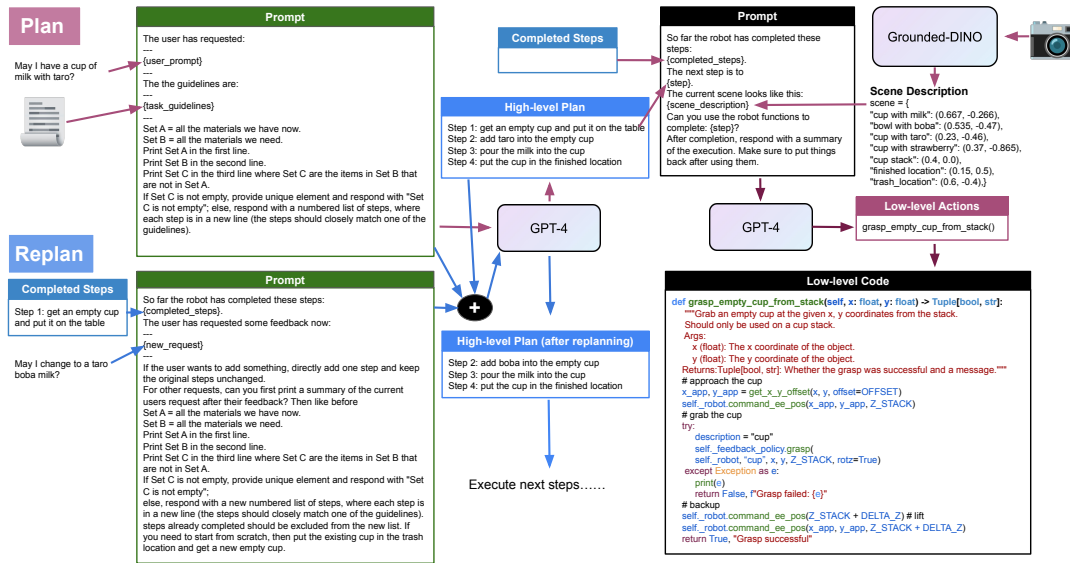


Figure 7.3: Detailed diagram of ITP with specific examples of user requests and task guidelines and replanning. During “Plan”: we feed user requests and task guidelines to complete the prompt and input it into GPT-4 to obtain a high-level plan. We input the history of completed steps and next step to prompt into the lower level executor to call the corresponding low-level actions. Once the lower level executor completes a step, we will maintain the history by storing it into *Completed Steps*. GPT-4 directly makes function calls to a predefined robot skill library (which could be learned or handcrafted). During “Replan”: we feed the completed steps and new request to create a new prompt, we append this new prompt to the previous conversation context and input the whole message into GPT-4 to obtain a new high-level plan. We refer this procedure as replanning, which previous language-based task planning methods have not considered. The low level executor then completes the next steps based on the new high-level plan.

7.4 Experiments

7.4.1 Robot Experiments

In our experiments, we focus on a drink-making system. Within the given scene, the robot is supplied with a set of ingredients that it must combine to produce a specific drink. Our setup also has an overhead camera that feeds images to Grounded-DINO model for scene understanding.

For the robot, we provide a predefined set of skills, which include actions like “grasp_cup”, “pour”, and “scoop_boba_to_location”. The “grasp_cup” skill is implemented with a feedback policy that centers the gripper on the cup, given the approximate location from the scene description, enabling the robot to grasp it reliably. The “pour” skill is designed to accept

User Request	Difficulty Level	Code as Policies		LCB	
		Planning	Success	Planning	Success
<i>I would like to order a cup of milk.</i>	Existed	3/3	✓	3/3	✓
<i>I want to order a boba milk.</i>	Existed	2/4	✗	4/4	✓
<i>Can I have a cup of strawberry milk?</i>	Existed	4/4	✓	4/4	✓
<i>I want a matcha latte.</i>	Zero-shot easy	4/4	✓	4/4	✓
<i>May I have a cup of milk with taro?</i>	Zero-shot easy	3/3	✓	3/3	✓
<i>I want taro milk with boba.</i>	Zero-shot moderate	3/5	✗	5/5	✓
<i>Can I get a strawberry boba milk?</i>	Zero-shot moderate	3/5	✗	5/5	✓
<i>I want to order a strawberry matcha milk.</i>	Zero-shot moderate	5/5	✓	5/5	✓
<i>I'd order a strawberry matcha milk with boba.</i>	Zero-shot hard	3/6	✗	6/6	✓
<i>I would like a cup of passion fruit milk.</i>	Unavailable material	-	✗	-	✓
Total	-	80%	5/10	100%	10/10

Table 7.1: Quantitative results with real robots for high-level planning rate and success rate with various user requests. For high-level planning, we extract planning accuracy by dividing the number of successful steps by the total number of steps, shown as ‘*Successful Steps / Total Steps*’. We determine success by whether the robot successfully accomplishes the task. To calculate the overall high-level planning score, we average the performance across all user requests.

a location and a descriptive cue of the ingredient being poured. This level of specification enables milk to be poured more than specific flavors. For example, when making a matcha latte, the *pour* function will be provided “*matcha*” or “*milk*” as inputs. When the input is “*matcha*”, the controllable tilt angle will be small, while when the input is “*milk*”, the controllable tilt angle will be much larger. This ensures that the robot can pour more milk and a bit of matcha liquid.

7.4.2 Comparison on Task Planning

We consider Code as Policies as a baseline. Code as Policies provides a formulation for language model-generated programs executed on real systems by prompting a text completion model with code examples. For a fair comparison, not only do we provide Code as Policies with the same information as given in ITP in the form of comments, but we also provide an additional 40 lines of code prompts providing example usage, as is done in Code as Policies. For both ITP and Code as Policies, we provide user requests and task guidelines as inputs. The task guidelines include 3 instances, along with their associated high-level planning steps, current available material, and other task-specific conditions. Our task guidelines are shown in Task Guidelines 7.1.

We evaluate the methods on two criteria: the number of high-level steps correctly generated and whether the real robot successfully finished the task. We send user requests of varying

complexity levels, including ‘existed’, ‘zero-shot easy’, ‘zero-shot moderate’, ‘zero-shot hard’ and ‘unavailable material’. ‘Zero-shot’ means the instruction for making the corresponding drink is not provided in the task guidelines. ‘Unavailable’ indicates that we don’t have the material for the requested beverage. We show the results in Table 7.1. We could notice that ITP is robust in high-level plan generation and can easily be generalized to novel instructions of unseen drinks or unavailable drinks. For example, the user sends the request ‘*I would like a cup of passion fruit milk.*’ However, passion fruit jam is not available, so the system will provide the response ‘*Passion fruit jam is not available*’ and stop the program. In comparison, Code as Policies failed to achieve this objective. To understand the failure case of Code as Policies, we provide some observations: 1) when making a cup of milk with boba, the system attempted to scoop boba from the working cup, improperly adhering to the correct usage of the lower-level skill. 2) When the prompt is more complex (9th row), the system adds milk first and then adds the boba, resulting in an incorrect execution order. 3) When the material is not available, it cannot justify that passion fruit doesn’t exist. Additionally, since ITP is built based on task guidelines alone, it demands significantly less prompt engineering than Code as Policies, which makes our system very easy to use for various task planning purposes.

7.4.3 Replan with Human-in-the-loop Feedback

Our system is robust to diverse new requests during execution. To verify this point, we assess the task replanning performance on real robots in response to a user’s new request, referred to as human-in-the-loop feedback. We display the results in Table 7.2. We notice that ITP demonstrates its capacity to effectively handle a range of new requests, even after progressing through various steps of the task. The last example is of particular note, where ITP adds one step more (‘Stir the mixture until the matcha powder is well mixed’) before putting the working cup in the finished location. Here the language model assumes the need to stir the matcha due to the ambiguity of the correct procedure. Such superfluous steps can be reduced by adding restrictions in the task guidelines, which can easily be done by a general user of the system. This contrasts with methods like Code as Policies which require tuning prompts at the code level.

7.4.4 Comparison on Simulation Tasks

In this section, we aim to verify ITP’s performance for simulation tasks. We compare our high-level planning module to that of ProgPrompt (Singh et al., 2023) by leveraging the simulated Virtual Home (VH) Environment (Puig et al., 2018). We would like to emphasize that ITP provides a user-friendly approach for users to input their high-level guidelines, which requires little background knowledge, while other approaches such as ProgPrompt employ a code-like prompting strategy. To make a fair comparison, we obtain high-level planning from both ITP and ProgPrompt, and use ProgPrompt’s low-level execution, strictly following the same evaluation protocol: each result is averaged over 5 runs in a single VH

User Request	New Request	Step When New Request is Made		
		1st	2nd	3rd
<i>Can I have a cup of strawberry milk?</i>	<i>I want to add boba into the drink.</i>	4/4	3/3	5/5
<i>I want a matcha latte.</i>	<i>Sorry, I want boba milk without matcha instead.</i>	3/3	5/5	5/5
<i>May I have a cup of milk with taro?</i>	<i>Can I replace the taro with strawberry?</i>	3/3	5/5	5/5
<i>Can I get a strawberry boba milk .</i>	<i>Sorry, can I reorder a strawberry milk?</i>	3/3	5/5	5/5
<i>A strawberry matcha milk with boba.</i>	<i>Can I just get matcha boba milk and no strawberry?</i>	4/4	<u>5/4</u>	<u>7/6</u>

Table 7.2: Replanning performance with real robots given human-in-the-loop feedback. After the user sends a request, we interrupt the procedure before different steps (1st, 2nd, and 3rd). Note that our replanning system is robust in handling these new requests. Interestingly, for the last example, after the 2nd and 3rd step, ITP adds one step more (‘Stir the mixture until the matcha powder is well mixed’) before putting the working cup in the finished location, leading to 5 and 7 steps instead of 4 and 6 steps respectively. We assume this is because GPT-4 assumes matcha powder is hard to mix, while we select water-soluble matcha powder. Including the instruction ‘matcha powder is water-soluble’ in the task guidelines could address this issue.

Environment across 10 different tasks. We display the results in Table 7.3. We notice that ITP is not only a user-friendly approach that allows users to provide high-level guidelines in more straightforward natural language but is able to match or exceed ProgPrompt on executable functions on the Virtual Home benchmark.

7.4.5 Discussion

Although many works (Singh et al., 2023; Liang et al., 2022; Skreta et al., 2023; Rana et al., 2023) have explored LLMs for understanding feedback and planning, none of these works both consider user-friendly task guidelines and replan based on a user’s new request. Prog-prompt (Singh et al., 2023) and Code as Policies focus more on making one plan and executing the task step by step, while also requiring complicated reference code. CLAIRIFY (Skreta et al., 2023) provides effective guidance to the language model by generating structured task plans and incorporating any errors as feedback, and SayPlan (Rana et al., 2023) introduces an iterative replanning pipeline that refines the initial plan using feedback from a scene graph simulator. However, these two approaches pay attention more on a task instead of users’ experience. Our ITP aims to provide a unified vision and language framework that can provide the best user experience, so the user with minimal specialized knowledge can still easily ask their robots to execute a task.

Task Description	$ A $	ProgPrompt	ITP
<i>watch tv</i>	3	0.42 ± 0.13	0.83 ± 0.06
<i>turn off light</i>	3	1.00 ± 0.00	0.75 ± 0.00
<i>eat chips on the sofa</i>	5	0.40 ± 0.00	0.96 ± 0.05
<i>brush teeth</i>	8	0.74 ± 0.09	0.86 ± 0.12
<i>throw away apple</i>	8	1.00 ± 0.00	1.00 ± 0.00
<i>make toast</i>	8	1.00 ± 0.00	0.59 ± 0.16
<i>put salmon in the fridge</i>	8	1.00 ± 0.00	1.00 ± 0.00
<i>bring coffeepot and cupcake to the coffee table</i>	8	1.00 ± 0.00	1.00 ± 0.00
<i>microwave salmon</i>	11	0.76 ± 0.13	0.89 ± 0.09
<i>wash the plate</i>	18	0.97 ± 0.04	0.95 ± 0.01
Avg: $0 \leq A \leq 5$		0.61 ± 0.29	0.84 ± 0.10
Avg: $6 \leq A \leq 10$		0.95 ± 0.11	0.90 ± 0.17
Avg: $10 \leq A \leq 18$		0.87 ± 0.14	0.92 ± 0.07

Table 7.3: Comparison of executability (Exec) on Simulation (Virtual Home) with ProgPrompt. Exec is the fraction of actions in the plan that are executable in the environment, even if they are not relevant for the task. ITP is not only a user-friendly approach that allows users to provide high-level guidelines, but it can also achieve superior results on varied tasks in the simulation.

7.5 Conclusion

Conclusion. In this chapter, we propose a simple yet effective system, ITP, which melds the capabilities of Large Language Models in an interactive system that constructs plans, and performs tasks centered around the users needs. Encouragingly, it precisely interprets user requests, generates pertinent step-by-step plans, and achieves the desired outcome — a testament to the potential of such systems for real-world applications. We embody our system in a robot designed to make various drinks according to user preferences and adeptly demonstrate its ability to respond to feedback during execution. Our system is capable in the context of interactive task planning and replanning for robotics.

Limitations and Future Work. While ITP provides a working proof of concept of an interactive robot system, there is room for enhancing its capabilities with more powerful robot skills to tackle more intricate tasks. Similarly, the integration of more precise visual information that leverages 3D information would significantly elevate the robot’s proficiency in understanding, planning, and interacting with its surroundings. We aim for our open-source system to inspire more research into using both established and emerging models to enhance real-world robotics.

Chapter 8

Hierarchical Policies with Language Models

This chapter is based on the paper “From LLMs to Actions: Latent Codes as Bridges in Hierarchical Robot Control” (Yide Shentu & Abbeel, 2024), by Yide Shentu, Philipp Wu, Aravind Rajeswaran, Pieter Abbeel.

Hierarchical control for robotics has long been plagued by the need to have a well defined interface layer to communicate between high-level task planners and low-level policies. With the advent of LLMs, language has been emerging as a prospective interface layer. However, this has several limitations. Not all tasks can be decomposed into steps that are easily expressible in natural language (e.g. performing a dance routine). Further, it makes end-to-end finetuning on embodied data challenging due to domain shift and catastrophic forgetting. We introduce our method – Learnable Latent Codes as Bridges (LCB) – as an alternate architecture to overcome these limitations. LCB uses a learnable latent code to act as a bridge between LLMs and low-level policies. This enables LLMs to flexibly communicate goals in the task plan without being entirely constrained by language limitations. Additionally, it enables end-to-end finetuning without destroying the embedding space of word tokens learned during pre-training. Through experiments on Language Table and Calvin, two common language based benchmarks for embodied agents, we find that LCB outperforms baselines (including those w/ GPT-4V) that leverage pure language as the interface layer on tasks that require reasoning and multi-step behaviors.

8.1 Introduction

The field of robotics has long oscillated between two predominant architectural paradigms for enabling agents to solve complex tasks. At one end of the spectrum, we have seen *modular hierarchical policies* (Liu et al., 2024a) for control that leverage rigid layers like symbolic planning, trajectory generation, and tracking. On the other end are *end-to-end*

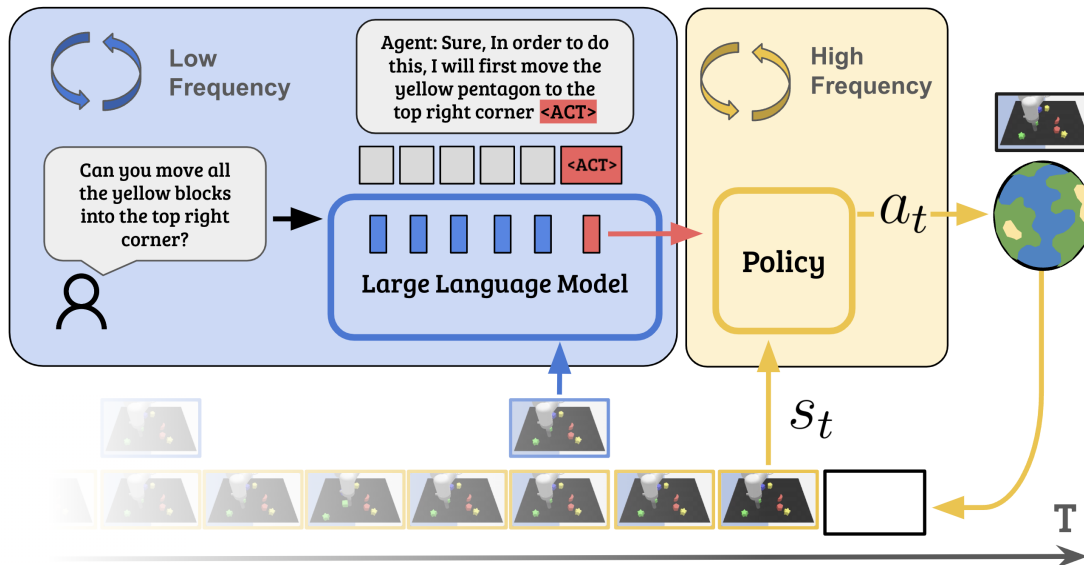


Figure 8.1: **Illustration of our proposed Latent Code as Bridges architecture.** Given a high-level task description and the observation, a Large Language Model (LLM) generates a textual description of an action and an <ACT> token. The feature embedding from the <ACT> token’s last layer serves as a high-level latent goal for the downstream policy network. Our modular hierarchical approach synergies the LLM’s high-level reasoning with the pre-trained policy’s responsive low-level control, addressing the limitations of direct action output by monolithic LLMs. Unlike methods that using a large LLM to directly output agent actions (Brohan et al., 2023), our approach can run the LLM reasoning and action policy execution loops asynchronously, mirroring human task execution with immediate low-level feedback when interacting with the physical world and slower, deliberate reasoning when considering longer term planning. At test time, the action policy frequently updates actions based on environment changes, while the LLM updates are less frequent, enabling efficient, real-world inference.

policies (Levine et al., 2016) that directly map sensory observations to actions through high-capacity neural networks. This dynamic history reflects the ongoing quest to reconcile the logical human-like reasoning with the flexible dexterity of human motor control.

The advent of *large language models* (LLMs) (OpenAI, 2023; Touvron et al., 2023) and their remarkable language interpretation and reasoning capabilities have reignited interest in hierarchical control architectures. Recent works (Ahn et al., 2022; Huang et al., 2022; Liang et al., 2022) have leveraged LLMs and *Multimodal Large Language Model* (abbreviated as LLM in this chapter unless specified otherwise) in place of high-level symbolic planners, enabling impressive results like mobile rearrangement of objects based on open-vocabulary instructions. Despite these advances, the core deficiencies of hierarchical architectures remain – namely the need for a set of clearly defined control primitives and an interface between layers in the hierarchy. For example, LLMs leverage the semantic meaning of action verbs to

coordinate low-level primitives like *go-to*, *pick*, *place* etc. However, we humans perform a variety of movements with our body that contribute to our dexterity and daily function, yet **cannot be easily described using language**.

In this backdrop, we present **Latent Codes as Bridges**, or **LCB**, a new policy architecture for control that combines the benefits of modular hierarchical architectures with end-to-end learning (see Fig. 8.1 for an illustration). Specifically, not only can **LCB** directly leverage LLMs for high-level reasoning and pre-trained skills/policies for low-level control, but it can also improve these components with end-to-end learning to transcend their initial capabilities. This is achieved by learning an `<ACT>` token at the interface layer which can modulate the low-level policies. As a result of this choice, **LCB** can overcome the inherent limitations of solely relying on language as the interface layer, since several behaviors are hard to describe in language. Secondly, by leveraging a separate `<ACT>` token, we do not destroy the core language generation and reasoning capabilities of the LLM during finetuning. We test **LCB** on a series of long-horizon and reasoning tasks in Language Table (Lynch et al., 2022) and Calvin (Mees et al., 2022), two common language based benchmarks for embodied agents. We find that **LCB** considerably outperforms baselines that leverage LLMs to sequence low-level skills using pure language as the interface layer. See our website for more.

8.2 Related Work

Hierarchical Control with LLMs The proliferation of LLM technology, coupled with their capability to interpret user prompts and perform reasoning, has led to growing interest in utilizing LLMs for robotics (Zeng et al., 2023; Vemprala et al., 2023). Of particular notice and relevance are the use of LLMs for high-level reasoning in hierarchical control architectures. Prior work has demonstrated this by leveraging the few-shot prompt capabilities of LLMs (Huang et al., 2022; Ahn et al., 2022), their ability to code and compose functions (Liang et al., 2022; Singh et al., 2023), or their ability to interact with human users through language (Wu et al., 2023c). In contrast to these works that attempt to use LLMs “as-is” and compose low-level skills, our work performs end-to-end fine-tuning through learnable latent codes. This includes finetuning some layers of the LLM through LoRA (Hu et al., 2022). Empirically we show that such finetuning can outperform methods that use LLMs out-of-the-box.

Language Conditioned Imitation Learning To leverage LLMs for task planning and reasoning, such models need to be able to call lower-level skills to affect change in the environment. This can be achieved in two ways: (a) by leveraging *semantics* of the skills through language descriptions (e.g. `go-to`, `reach` etc.) as described above; or alternatively (b) through language conditioned policies which accept a text description as input to directly produce an action (Lynch et al., 2022; Jang et al., 2022; Brohan et al., 2023; Li et al., 2023b; Lynch & Sermanet, 2021). Such policies can typically perform only short horizon tasks and lack the reasoning and planning capabilities often found in LLMs. Our goal in this work is to leverage such “simple” or “primitive” language-conditioned policies along with LLMs to

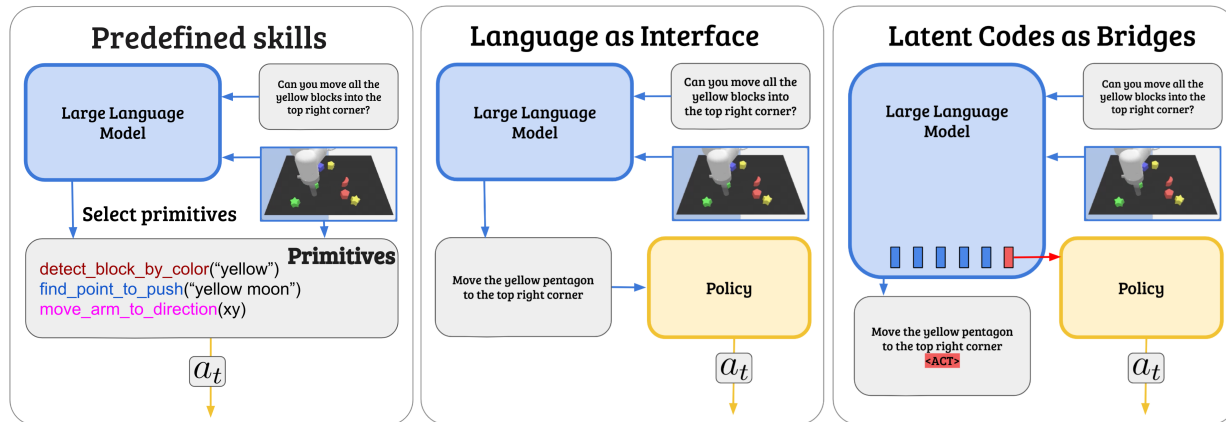


Figure 8.2: A high level architectural comparison of LLM-based hierarchical policies. Predefined skills (left) uses a LLM to call predefined primitives. Language as an interface (middle) uses a LLM to output a simple language command, which is then passed into a language conditioned policy. **LCB** (right) utilizes a latent code as a **bridge** between the LLM and the low level policy, facilitating hierarchical control and end-to-end learning.

enable a hierarchical system to perform complex tasks that require multi-step planning and reasoning.

Large Pre-Trained Models for Embodied Agents Recent years have witnessed growing interest in robotics to re-use large models originally trained for vision or language applications (Kirillov et al., 2023; Vemprala et al., 2023) or their architectures (Chen et al., 2021; Janner et al., 2021; Wu et al., 2023d; Liu et al., 2023a). We are also starting to see large models and representations custom trained for robotics (Brohan et al., 2023; Nair et al., 2022; Majumdar et al., 2023; Yang et al., 2024). In our work, we leverage the recent class of Multimodal Large language models (Liu et al., 2023b; Zhu et al., 2023; Li et al., 2023a) that extend the capability of text only LLMs to interpret other modalities like vision through alignment layers. Specifically, our instantiation of LCB model builds on top of LLaVA (Liu et al., 2023b) and finetunes the model on a simulated dataset of embodied reasoning and long-horizon tasks. As the availability of embodied datasets paired with language annotations grow, we hope that our method can be extended to release generalist models that can be deployed zero shot in new domains.

8.3 Method

We wish to develop a hierarchical policy architecture that can enable robots to perform a variety of manipulation tasks when provided with free-form language descriptions. Specifically, we seek an architecture that can handle low-level actions for fine-grained or contact-rich

tasks (e.g. pushing, 6D object manipulation) while also having the capability to reason and plan without any external step-by-step instructions. Before we present our architecture for this purpose, we first survey two other families of approaches and their deficiencies, which provides the intuition and basis for our method. These approaches are shown in Figure 8.2.

LLMs Leveraging Predefined Skills First we can consider a hierarchical approach where LLMs perform high-level task planning by calling a set of pre-defined skills or APIs (Ahn et al., 2022; Liang et al., 2022). These APIs (e.g. `go-to`, `push`) are described and provided to the LLM as part of the main prompt. This approach suffers from two primary drawbacks. Firstly, for an LLM to plan with skills, they need to have *semantics* attached to them that make linguistic sense. Secondly, this constrains the set of skills to a closed vocabulary, and prevents any form of generalization to new skills or capabilities. Furthermore, code-writing proficiency demands a high-quality LLM, a criterion met chiefly by proprietary commercial models such as GPT-4 (Liu et al., 2024a). Additionally, end-to-end fine-tuning is challenging since the LLM cannot adapt or compensate for limited prowess of the low-level skills (Ahn et al., 2022).

Language as Interface The second class of approaches can leverage *language-conditioned low-level policies* as opposed to a finite set of low-level skills. Such policies can take a simple language command as input (e.g. `pickup the red block`) and produce actions that can (hopefully) accomplish the task. Since these policies can accept free-form text as input, at least theoretically, they have the capability to generalize to new instructions. Furthermore, they are amenable to end-to-end fine-tuning from high-level instructions, through an LLM, to the language conditioned policy, and ultimately the action. Nevertheless, this class of approaches also suffer from key limitations. Firstly, not all high level tasks can be decomposed into sub-tasks in simple language. For example, imagine trying to describe step-by-step instructions to make a robot dance to a song. Secondly, end-to-end fine-tuning with such an architecture can destroy planning and reasoning capabilities that the LLM originally had (Luo et al., 2023).

Latent Codes as a Bridge (Ours) Finally we describe our method which can overcome the key limitations outlined above. Our key insight is that we can introduce an additional latent code to act as a bridge between the high-level LLM and low-level language conditioned policy. We augment the LLM’s tokenizer by adding a specialized `<ACT>` token, prompting the model to predict this token in response to actionable questions. The last layer embedding of the `<ACT>` token is then utilized as a latent goal for the downstream policy network. This learnable `<ACT>` token’s embedding facilitates the transmission of abstract goals and nuances to the low-level policy – details that are not easily conveyed through language alone. Furthermore, by using this additional learnable token, we preserve the embedding space for language tokens, thus preventing any catastrophic forgetting during end-to-end fine-tuning. We describe more specific details of our architecture and implementation below.

8.3.1 Architecture and Implementation Details of LCB

LCB unifies the capabilities of a slow but powerful pretrained Multimodal Large Language Models (LLMs) with a fast and simpler decision-making policies to create a model that ingests vision and language inputs to output low-level actions. This integration involves a two-component system: a pretrained LLM, denoted as f_ϕ , and a pretrained policy, π_θ , parameterized by ϕ and θ respectively. The LLM consists of a text only large language model and a vision encoder, which projects images into the text only large language models embedding space, facilitating a multimodal understanding of textual and visual inputs. In this work, we leverage LLaVA (Liu et al., 2023b) as our pretrained LLM. f_ϕ takes in text tokens x_{txt} and images x_{img} and outputs text tokens. The pretrained policy π_θ takes as input environment observations at the current time step o_t , with conditioning latent z , and outputs the action at the current time step a_t .

We introduce an additional `<ACT>` token into the vocabulary of the language model, which is a special token that enables the language model to generate an action embedding to control the lower level policy. The model is trained to output `<ACT>` tokens when executable requests are provided to the model. We extract out the last-layer embedding features from the model of at the `<ACT>` token, following the approach used in Language Instructed Segmentation Assistant (LISA) (Lai et al., 2023). This embedding is projected into the policy latent conditioning space by a linear layer to extract the latent feature $z_{\text{<ACT>}}$ which is then fed into the policy π_θ .

8.3.2 Data Processing

The LCB framework necessitates diverse and strategically curated datasets to make the policy effective for language-guided action execution in varied contexts. We cater the data collection and preprocessing steps towards this goal, creating a small instruction tuning dataset.

We convert in domain text conditioned policy data into the chat format of LLM assistants. Typical language conditioned trajectory datasets contain one language instruction and a list of (observation, action) pairs $[x_{txt}, (o_0, a_0, \dots, o_t, a_t, \dots)]$ per trajectory. We programmatically generate text data in the format of chat interactions using templates. A simple example of this user-assistant interaction, is “User: can you help me x_{txt} ? Assistant: yes, `<ACT>`.” Specific templates for chat data generation are provided in Appendix E.2. This trains the model to recognize and respond to direct action requests, fostering a conversational interface that seamlessly transitions from dialogue to action.

Moreover, we enrich our training material with additional datasets designed to prompt specific behaviors from the language model. One such data source is reasoning data, where the model is tasked with a more abstract goal and must reason about the scene to accomplish the goal. Such examples are framed within a chat-like interaction, encouraging the model to articulate its reasoning process before executing the `<ACT>` command. For example, “User: x_{img} Can you x_{txt} ? Assistant: I will x_{goal} `<ACT>`”. Where x_{txt} does not explicitly specify the target object and location. If x_{txt} is “move the block closest to the bottom right to the block

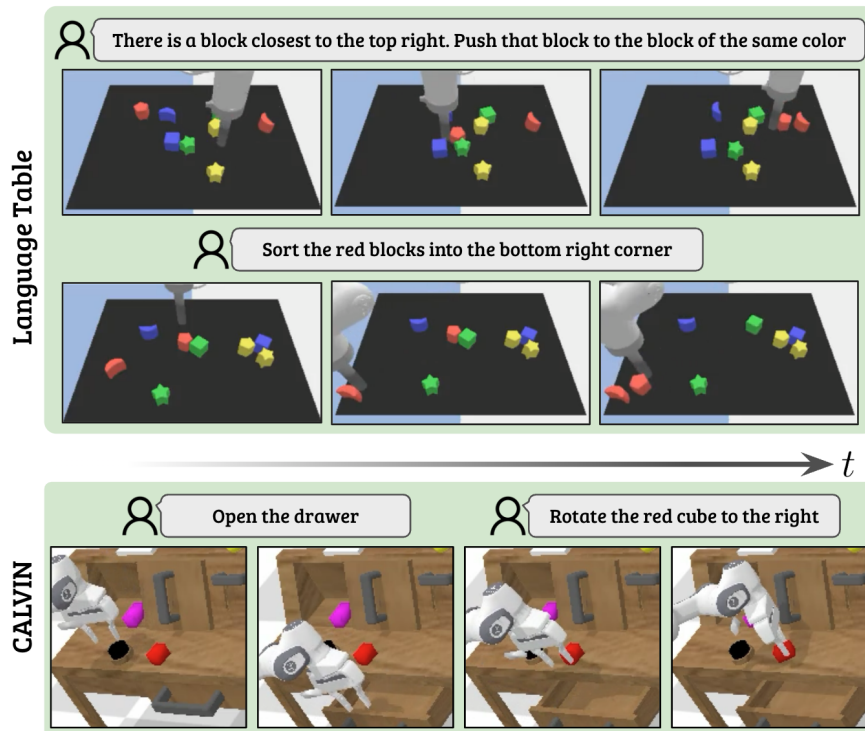


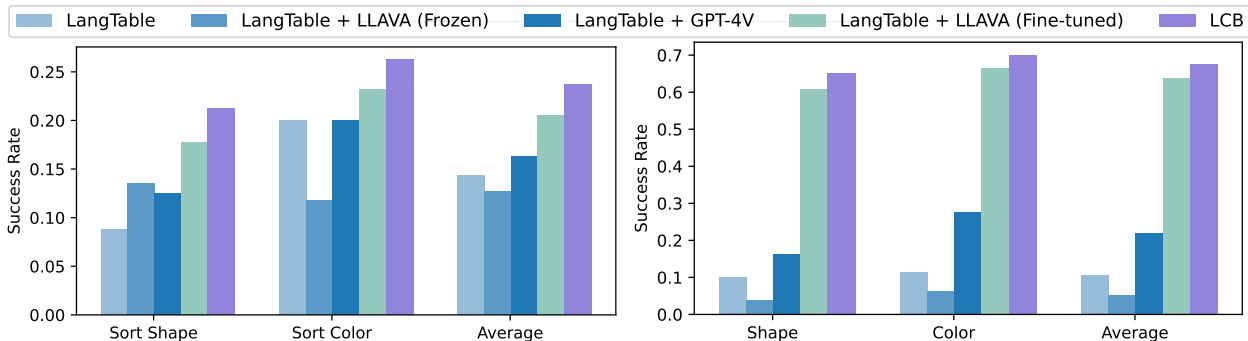
Figure 8.3: A visualization of the two environments along with exemplar tasks that we train and evaluate on. The top depicts the Language Table environment (Lynch et al., 2022). We study reasoning tasks (first trajectory) and long horizon tasks (second trajectory). The bottom depicts the CALVIN long horizon benchmark (Mees et al., 2022), in which the agent must sequentially accomplish tasks.

of a similar color”, the assistant’s response, x_{goal} , provides an explanation of the task, such as “I will move the blue cube on the bottom right to the blue moon”.

We also study long-horizon tasks and incorporate training sequences that require the model to plan and execute multiple steps to achieve a goal. This is achieved by defining task stages (start, regular, transition, stop) and incorporating the previous action as context in the language model’s input. This strategy trains the model to recognize task progression and adapt its actions accordingly, enabling it to manage tasks with evolving objectives. Through this dataset strategy, our model is finely tuned as a versatile tool capable of understanding and executing a wide range of language-guided actions.

8.3.3 Training

The training of LCB employs a combination of techniques to integrate the LLM and policy components. We leverage Low Rank Adaptation (Hu et al., 2022) (LoRA) for fine-tuning the LLM, allowing for more efficient training. We adopt a cold start approach to policy



(a) **Long Horizon** Success rate for the multi-step tasks on Language Table. The task requires sorting blocks based on color or shape. The environment only provides the high level objective to each method. This task requires the policy to have more long term planning capabilities, whether explicitly or implicitly.

(b) **Reasoning:** Success rate for the reasoning tasks on Language Table. The reasoning task is specified as a variant of "There is a block that is closest to i.e., top right corner. Push that block to the other block of the same shape/color." This task requires the agent to understand object semantics and spacial relationships.

Figure 8.4: Task success rates on Language table. The tasks are drawn from the higher level Language Table tasks from PALM-E (Driess et al., 2023). LangTable refers to the original language table policy (Lynch et al., 2022). +LLaVA (frozen) refers to composing the original language table with a frozen LLaVA model and few shot prompting. +GPT-4V similarly refers to composing the original policy with GPT-4V. +LLaVA (finetuned) refers to finetuning the LLaVA policy on our mixture dataset on the language only, then composing it with the policy. Our results show that leveraging LCB is effective on tasks that require additional reasoning and planning. Note that the same model is evaluated between the long horizon and reasoning tasks.

training, reminiscent of staged training strategies seen in prior works, by first freezing the action decoder and only fine-tuning the language model. This preliminary phase focuses on aligning the embeddings produced by the LLM with the feature space of the policy. We find that adding an additional CLIP loss to regularize the latent embedding $z_{\langle \text{ACT} \rangle}$ is necessary, ensuring that the embeddings from the language model remain well aligned with the lower level ground truth text description g_{txt} of the objective for the pre-trained policy. In total, our loss function is comprised of 3 terms, and can be expressed as follows:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{policy}}(\pi_\theta, o_t, a_t, z_{\langle \text{ACT} \rangle}) \quad (8.1)$$

$$+ \lambda_2 \mathcal{L}_{\text{LM}}(f_\phi, x_{\text{txt}}, x_{\text{img}}) \quad (8.2)$$

$$+ \lambda_3 \mathcal{L}_{\text{CLIP}}(z_{\langle \text{ACT} \rangle}, g_{\text{txt}}) \quad (8.3)$$

Table 8.1: Comparison on the original Language Table benchmark tasks. LangTable is the original language table policy Lynch et al. (2022). LCB is our method applied only to the original Language Table dataset. We see that LCB can help improve task performance by leveraging the vision language model for feature extraction. The tasks are: Block to Block (B2B), Block to Block Relative Location (B2RL), Seprate (S), Block to Relative Location (B2RL), and Block to Absolute Location (B2AL).

Model	B2B	B2BRL	S	B2RL	B2AL	<i>Avg</i>
LangTable	0.88	0.70	0.94	0.68	0.65	0.77
LCB	0.90	0.66	0.99	0.73	0.71	0.80

8.4 Results

We systematically evaluated LCB across a diverse set of environments and tasks to demonstrate the efficacy of integrating a pretrained Large Language Model (LLM) with a domain-specific, pretrained low-level policy. Our primary objective was to study the capabilities of the policy, specifically its high-level language understanding and low-level control. Through our experiments, we aim to answer the following questions:

- Does LCB enable learning a bridge between the LLM and the policy more effectively than pure language?
- Can LCB leverage the pretrained capabilities of LLMs to solve long horizon tasks by decomposing the high level goals into the step by step latent commands?
- Can LCB outperform other baseline methods that leverage close-sourced state of the art LLMs such as GPT-4V?

To answer these questions, we study how LCB performs under various reasoning and long horizon settings in both the Language Table and CALVIN benchmarks. See Figure 8.3 for a visualization of the environments and example tasks.

8.4.1 Evaluation on Language Table

Language Table offers a simulated tabletop environment for executing language-conditioned manipulation tasks (Lynch et al., 2022). The environment features a flat surface populated with blocks of various colors and shapes, alongside a robot with 2D action space. Language Table provides observations in the form of the robot end effector position and third-person camera images. Despite its simplicity, it provides a reproducible and comprehensive environment to study challenges at the interface of high level language and low level contact-rich dynamics and feedback control.

We investigate the benefit of using LCB on the original Language Table benchmark. Here we apply our method using the same dataset that the original language table model was trained on, translating the original language instructions into chat interactions with action tokens as specified in section 8.3. As shown in Table 8.1, with the end to end optimization with the pretrained LLM, the success rate across the benchmark matches or exceeds the baseline Language Table approach. This signifies that LCB is able to seamlessly adapt a pretrained LLM and policy together. We suspect that this is due to the flexibility in the latent representation $z_{\langle \text{ACT} \rangle}$, allowed for by our approach as well as additional capacity afforded by the language model.

We next investigate more complex language tasks that require reasoning and planning capabilities. We collect a small dataset for each capability, training models to compare the following approaches:

- **LangTable:** The original Language Table Policy, as provided by (Lynch et al., 2022).
- **LangTable + LLaVA (Frozen):** The combination of the original policy and a non-fine-tuned LLaVA model interfacing through language. We prompt LLaVA to output language commands in the format and style as expected by LangTable.
- **LangTable + GPT-4V:** The integration of LangTable with the state-of-the-art proprietary Vision Language Model (GPT-4V). In order to bootstrap the spatial understanding of GPT-4V, we also incorporate the Set of Marker (SOM) (Yang et al., 2023) technique to enhance the GPT-4V’s capability. We further include multi-modal few shot contexts including language explanation of the tasks and image examples. More details are provided in Appendix E.1.
- **LangTable + LLaVA (Fine-tuned):** The original policy augmented by a LLaVA model that has been fine-tuned on the exact language needed for the action policy for the given task.
- **LCB:** We take a pretrained LLaVA model and the pre-trained LangTable policy and apply LCB, learning a latent interface between the two on the respective instruction dataset.

Results for long horizon performance are provided in Figure 8.4a. In this task, the agent must sort blocks based on shape or color into a specified corner of the board, requiring a long sequence of actions from which the agent could greatly benefit through high-level planning. We see that LCB exhibits a competency for handling such tasks, as indicated by the heightened success rates, improving on pure language interface baselines. This is attributable to the method’s ability to generate a coherent sequence of latent action embeddings that guide the policy through the task’s duration, facilitating a more consistent and accurate alignment with the sequential nature of the task. During evaluation we run the higher level language model at a slower rate than the lower level policy, only updating the language models output

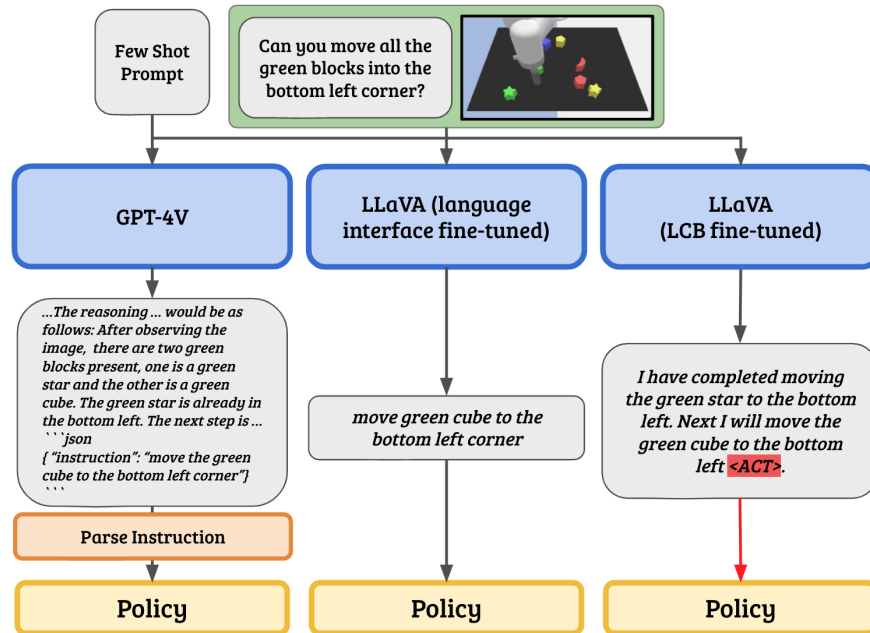


Figure 8.5: A comparison of the flow from a high level language task to the policy for different approaches. **(Left) LangTable + GPT-4V** requires a prompt to understand the task and desired output format. GPT-4V can provide language reasoning to allow the user to introspect the decision process of the language model, but requires additional parsing to extract the relevant language instruction to provide to the model. **(Middle) LangTable + LLaVA (Fine-tuned)** fine-tunes the language model to output the exact language instruction as in the training data, effectively acting as a language interface converter. This approach, while effective, removes the chat like capability from the language model. **(Right) LCB** fine-tunes the language model with a chat like interface and action token. The policy is directly conditioned on the latent feature from the action token provided by the model, enabling effective policy conditioning without losing the chat like language model interface.

every 40 environment steps. We find that this increases computational efficiency without compromising task performance suggesting the effectiveness of the model hierarchy.

Results for reasoning performance are provided in Figure 8.4b. Tasks here are of the form “There is a block that is closest to {corner}. Push that block to the other block of the same {shape/color}”. In order to successfully accomplish this task, the agent must identify which block is located closest to a given corner, identify the relevant property (i.e. shape or color) and consolidate that understanding into an executable instruction. We see that our approach is able to outperform baselines that involve zero-shot prompting as well as naively fine-tuning the language model to output the translated robot task. We see that fine-tuning the language model to output the ground truth language primitive is effective in reaching parity with the oracle language baseline, but that LCB is able to match and even exceed that.

We provide a qualitative assessment of the language output from the various top performing approaches in Figure 8.5. LangTable + GPT-4V requires heavy prompt engineering and additional string parsing to extract out the final policy. LangTable + LLaVA is effectively fine-tuned by outputting the direct low level text command to the policy, but no longer is able to maintain a chat like interface to the user. In contrast, LCB is able to output an effective embedding for the low level policy while also verbalizing its reasoning. This decouples the low level policy conditioning from the language models text outputs, offering increased flexibility during instruction fine-tuning.

8.4.2 Evaluation on CALVIN

CALVIN(Mees et al., 2022) is an open-source simulated benchmark designed for learning long-horizon tasks conditioned by language. The environment features a 7-DOF Franka Emika Panda robotic arm equipped with a parallel gripper, situated at a desk with a variety of articulated furniture and objects for interaction. In each experiment, the robot needs to solve a sequence of complex full 6D manipulation tasks governed by real-world physics and guided by a series of language instructions. Each subtask is paired by a specific language instruction; upon successful completion, the robot proceeds to the next subtask accompanied by a new instruction. CALVIN encompasses four distinct environments A, B, C and D, with a shared set of language instructions and subtasks.

Table 8.2: Task completion rates for various methods on CALVIN(Mees et al., 2022) long-horizon tasks. All methods were trained exclusively on the ABC split of Calvin with the original language annotations and tested on split D with GPT-4 enriched language annotations, following the RoboFlamingo enriched instruction evaluation setting(Liu et al., 2024b). *RF denotes our own training of the RoboFlamingo model on the ABC Calvin split. 3DDA denotes the policy from 3D Diffuser Actor (Ke et al., 2024).

	Model	RF(Li et al., 2023b)	3DDA(Ke et al., 2024)	LCB
Task Completed in a Sequence (Success Rate)	1/5	0.620	0.652	0.736
	2/5	0.330	0.391	0.502
	3/5	0.164	0.203	0.285
	4/5	0.086	0.117	0.160
	5/5	0.046	0.061	0.099
	Avg Len	0.40	1.42	1.78

In order to demonstrate the generalization capabilities of LCB cross various environments as well as its ability to comprehend and act upon the same instructions phrased differently in

the CALVIN long horizon full 6D manipulation setting, we compare the following approaches:

- **RoboFlamingo (RF):** RoboFlamingo(Li et al., 2023b) adapts OpenFlamingo(Awadalla et al., 2023) by fine-tuning solely the cross-attention layer to directly output actions, thus maintaining its language comprehension. However, this approach requires executing the entire LLM anew with each progression to a subsequent state, leading to inefficiencies.
- **3D Diffusion Actor (3DDA):** Incorporating a diffusion policy with 3D scene representation and CLIP(Radford et al., 2021) language embedding, the 3D Diffusion Actor (Ke et al., 2024) sets the current SOTA on the Calvin benchmark when provided with standard language instruction inputs. However, a notable limitation stems from the constraints of the CLIP text model it employs. 3DDA can not generalize well on language instruction outside of its training distribution.
- **LCB:** LCB for Calvin integrates a pre-trained LLaVA(Liu et al., 2023b) as the Multimodal Large Language Model backbone with a pre-trained 3D Diffusion Actor serving as the action policy. This combination leverages the SOTA capabilities of the 3D Diffusion Actor to achieve a synergistic effect: LCB for Calvin excels in both language comprehension and low-level manipulation. Since RoboFlamingo runs the entire LLM on every environment step, in order to make a fair comparison, we also run the LLM part of LCB synchronously with the downstream policy, although we notice no significant performance difference for Calvin.

Table 8.2 presents results for the CALVIN long-horizon, language-conditioned benchmark. In this setting, the robot executes a series of tasks in unfamiliar environments based on novel GPT-4 enriched(Awadalla et al., 2023) instructions not encountered during training. The experimental outcomes demonstrate our approach’s distinct advantage over baseline methods. LCB significantly surpasses all baselines in terms of task success rate at every stage and in average completed trajectory length.

8.5 Conclusion

In this chapter, we introduce a novel approach, Latent Codes as Bridges, or LCB, that combines the abstract reasoning capabilities of large language models with low-level action policies. Our methodology does not merely stack these capabilities as in prior works but integrates them in an end-to-end fashion through a learned latent interface. The empirical evidence from our evaluations on the Language Table and CALVIN benchmarks shows the model’s adeptness in interpreting and executing various reasoning and long horizon objectives. The flexibility and effectiveness of the hierarchy enabled by LCB shows promise for real world robotic applications.

Chapter 9

Conclusion

This dissertation explores developing efficient learning methods for robotics, focusing on leveraging trajectory data of real-world robotic systems. By investigating representation learning, data collection strategies, policy learning frameworks, and the role of language models in robotics, this dissertation contributes to the advancement of scalable general-purpose robotic systems.

We started by exploring trajectory representation learning in Chapters 2 and 3. By introducing Masked Trajectory Models (MTM) and Semi-Supervised One-Shot Imitation Learning, this dissertation demonstrates how meaningful representations can be learned from trajectory data. These representations can facilitate a variety of downstream tasks, such as return-conditioned behavior cloning, trajectory clustering, and inverse dynamics. We also explore how to learn such representations from heterogeneous and multi-modal datasets effectively. These findings highlight the important role of trajectory representations in the advancement of robot learning systems.

We then address the practical real-world challenges of trajectory data collection and policy learning in Chapters 4, 5, and 6. We introduce DayDreamer as a generic method for sample-efficient policy learning in the real world by leveraging world models. Next, we explore GELLO as a teleoperation device to enable scalable, high-quality data collection for manipulation. RoboCopilot then further improves the applicability of GELLO by integrating iterative feedback into policy learning, demonstrating how human-in-the-loop strategies can enhance performance while targeting failure modes. These contributions emphasize the importance of quality data collection and sample efficient algorithms.

Finally, Chapters 7 and 8 examine the coordination of low-level control with high-level reasoning, integrating modular architectures with large language models. The methods employed in Interactive Task Planning and Latent Code as Bridges illustrate how vision language models can be combined with control policies to bridge the gap between abstract planning and precise execution. This hierarchical approach highlights the potential of leveraging high-level reasoning to inform low-level control, advancing the field of adaptive and interactive robotic systems.

Together, these contributions present a cohesive pathway toward scalable and general-

purpose robotics. Looking ahead, the findings in this dissertation inspire confidence in the development of robotic systems capable of operating autonomously in unstructured, human-centric environments. Continued progress will require addressing challenges such as refining the interaction between high-level reasoning and low-level execution and scaling these systems to diverse real-world applications. Additionally, having scalable methods that enable improving *beyond* human imitation also remains an open problem. As robotics continues to benefit from advances in hardware and algorithmic sophistication, the methods and frameworks presented here serve as a foundation for future research and application.

This work represents an important step toward the realization of robotic systems that go beyond task-specific solutions to become integral components of human life. I hope the insights gained herein will contribute to shaping the next generation of robotics, laying the groundwork for systems that are adaptive, efficient, and capable of meeting the complex demands of the real world.

Bibliography

- Abbeel, P. and Ng, A. Apprenticeship learning via inverse reinforcement learning. *Proceedings of the twenty-first international conference on Machine learning*, 2004.
- Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., Yan, M., and Zeng, A. Do as I can, not as I say: Grounding language in robotic affordances, April 2022.
- Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. Solving rubik’s cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.
- Albu-Schäeffler, A., Haddadin, S., Ott, C., Stemmer, A., Wimböck, T., and Hirzinger, G. The dlr lightweight robot – design and control concepts for robots in human environments. *INDUSTRIAL ROBOT-AN INTERNATIONAL JOURNAL*, 34:376–385, 08 2007. doi: 10.1108/01439910710774386.
- Amini, A., Mirbagheri, A., and Homayoun Jafari, A. Bilateral control of a nonlinear teleoperation robotic system with time varying delay using optimal control method. In *2017 24th National and 2nd International Iranian Conference on Biomedical Engineering (ICBME)*, pp. 330–333, 2017.
- Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009. ISSN 0921-8890.
- Arunachalam, S. P., Güzey, I., Chintala, S., and Pinto, L. Holo-dex: Teaching dexterity with immersive mixed reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5962–5969. IEEE, 2023.
- Åström, K. and Murray, R. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2010. ISBN 9781400828739.

- Awadalla, A., Gao, I., Gardner, J., Hessel, J., Hanafy, Y., Zhu, W., Marathe, K., Bitton, Y., Gadre, S., Sagawa, S., Jitsev, J., Kornblith, S., Koh, P. W., Ilharco, G., Wortsman, M., and Schmidt, L. Openflamingo: An open-source framework for training large autoregressive vision-language models, 2023.
- Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization, 2016.
- Bacchus, F. and Yang, Q. The downward refinement property. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'91, pp. 286–292, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc. ISBN 1558601600.
- Bahl, S., Gupta, A., and Pathak, D. Human-to-robot imitation in the wild. *RSS*, 2022.
- Baker, B., Akkaya, I., Zhokhov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *ArXiv*, abs/2206.11795, 2022.
- Bao, H., Dong, L., and Wei, F. Beit: Bert pre-training of image transformers. *ArXiv*, abs/2106.08254, 2021.
- Barbieri, L., Bruno, F., Gallo, A., Muzzupappa, M., and Russo, M. L. Design, prototyping and testing of a modular small-sized underwater robotic arm controlled through a master-slave approach. *Ocean Engineering*, 158:253–262, 2018. ISSN 0029-8018.
- Becker-Ehmck, P., Karl, M., Peters, J., and van der Smagt, P. Learning to fly via deep model-based reinforcement learning. *arXiv preprint arXiv:2003.08876*, 2020.
- Bertsekas, D. *Dynamic Programming and Optimal Control*. Number v. 1 in Athena Scientific optimization and computation series. Athena Scientific, 1995. ISBN 9781886529120.
- Bharadhwaj, H., Babaeizadeh, M., Erhan, D., and Levine, S. Information prioritization through empowerment in visual model-based rl. *arXiv preprint arXiv:2204.08585*, 2022.
- Billard, A. G., Calinon, S., and Guenter, F. Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems*, 54(5):370–384, 2006. ISSN 0921-8890. The Social Mechanisms of Robot Programming from Demonstration.
- Bohez, S., Tunyasuvunakool, S., Brakel, P., Sadeghi, F., Hasenclever, L., Tassa, Y., Parisotto, E., Humplik, J., Haarnoja, T., Hafner, R., Wulfmeier, M., Neunert, M., Moran, B., Siegel, N., Huber, A., Romano, F., Batchelor, N., Casarini, F., Merel, J., Hadsell, R., and Heess, N. Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors, 2022.
- Bonet, B. and Geffner, H. Planning as heuristic search. *Artificial Intelligence*, 129(1):5–33, 2001.

- Bousmalis, K., Vezzani, G., Rao, D., Devin, C., Lee, A. X., Bauza, M., Davchev, T., Zhou, Y., Gupta, A., Raju, A., Laurens, A., Fantacci, C., Dalibard, V., Zambelli, M., Martins, M., Pevcevičute, R., Blokzijl, M., Denil, M., Batchelor, N., Lampe, T., Parisotto, E., Żołna, K., Reed, S., Colmenarejo, S. G., Scholz, J., Abdolmaleki, A., Groth, O., Regli, J.-B., Sushkov, O., Rothörl, T., Chen, J. E., Aytar, Y., Barker, D., Ortiz, J., Riedmiller, M., Springenberg, J. T., Hadsell, R., Nori, F., and Heess, N. Robocat: A self-improving foundation agent for robotic manipulation, 2023.
- Brantner, G. and Khatib, O. Controlling ocean one: Human–robot collaboration for deep-sea manipulation. *Journal of Field Robotics*, 38(1):28–51, 2021.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.
- Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jackson, T., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, K.-H., Levine, S., Lu, Y., Malla, U., Manjunath, D., Mordatch, I., Nachum, O., Parada, C., Peralta, J., Perez, E., Pertsch, K., Quiambao, J., Rao, K., Ryoo, M., Salazar, G., Sanketi, P., Sayed, K., Singh, J., Sontakke, S., Stone, A., Tan, C., Tran, H., Vanhoucke, V., Vega, S., Vuong, Q., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. Rt-1: Robotics transformer for real-world control at scale, 2022.
- Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., Florence, P., Fu, C., Arenas, M. G., Gopalakrishnan, K., Han, K., Hausman, K., Herzog, A., Hsu, J., Ichter, B., Irpan, A., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, L., Lee, T.-W. E., Levine, S., Lu, Y., Michalewski, H., Mordatch, I., Pertsch, K., Rao, K., Reymann, K., Ryoo, M., Salazar, G., Sanketi, P., Sermanet, P., Singh, J., Singh, A., Soricut, R., Tran, H., Vanhoucke, V., Vuong, Q., Wahid, A., Welker, S., Wohlhart, P., Wu, J., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- Buamane, T., Kobayashi, M., Uranishi, Y., and Takemura, H. Bi-act: Bilateral control-based imitation learning via action chunking with transformer, 2024.
- Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

- Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.
- Carroll, M., Paradise, O., Lin, J., Georgescu, R., Sun, M., Bignell, D., Milani, S., Hofmann, K., Hausknecht, M. J., Dragan, A. D., and Devlin, S. Unimask: Unified inference in sequential decision problems. *ArXiv*, abs/2211.10869, 2022.
- Celemin, C., Pérez-Dattari, R., Chisari, E., Franzese, G., de Souza Rosa, L., Prakash, R., Ajanović, Z., Ferraz, M., Valada, A., Kober, J., et al. Interactive imitation learning in robotics: A survey. *Foundations and Trends® in Robotics*, 10(1-2):1–197, 2022.
- Chebotar, Y., Molchanov, A., Behtle, S., Righetti, L., Meier, F., and Sukhatme, G. S. Meta learning via learned loss. *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 4161–4168, 2021.
- Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In *Neural Information Processing Systems*, 2021.
- Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.
- Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.
- Chi, C., Xu, Z., Pan, C., Cousineau, E., Burchfiel, B., Feng, S., Tedrake, R., and Song, S. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.
- Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pp. 4754–4765, 2018.
- Connexion, D. Spacemouse, 2023. URL <https://3dconnexion.com/us/spacemouse/>.
- Crooks, W., Vukasin, G., O’Sullivan, M., Messner, W., and Rogers, C. Fin ray® effect inspired soft robotic gripper: From the roboSoft grand challenge toward optimization. *Frontiers in Robotics and AI*, 3, 2016. ISSN 2296-9144. doi: 10.3389/frobt.2016.00070. URL <https://www.frontiersin.org/articles/10.3389/frobt.2016.00070>.
- Das, N., Behtle, S., Davchev, T., Jayaraman, D., Rai, A., and Meier, F. Model-based inverse reinforcement learning from visual demonstrations. In *CoRL*, 2020.

- Dasari, S. and Gupta, A. Transformers for one-shot visual imitation. In *Conference on Robot Learning*, pp. 2071–2084. PMLR, 2021.
- Dasari, S., Ebert, F., Tian, S., Nair, S., Bucher, B., Schmeckpeper, K., Singh, S., Levine, S., and Finn, C. Robonet: Large-scale multi-robot learning, 2019.
- Dass, S., Pertsch, K., Zhang, H., Lee, Y., Lim, J. J., and Nikolaidis, S. Pato: Policy assisted teleoperation for scalable robot data collection. *RSS*, 2023.
- Dass, S., Ai, W., Jiang, Y., Singh, S., Hu, J., Zhang, R., Stone, P., Abbatematteo, B., and Martin-Martin, R. Telemoma: A modular and versatile teleoperation system for mobile manipulation. *arXiv preprint arXiv:2403.07869*, 2024.
- Deisenroth, M. P., Neumann, G., Peters, J., et al. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1–2):1–142, 2013.
- Deng, F., Jang, I., and Ahn, S. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations. *arXiv preprint arXiv:2110.14565*, 2021.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- Dimension, F. Haptic devices, 2023. URL <https://www.forcedimension.com/products>.
- Donnelly, K. Heteromodal cortex. In *Encyclopedia of Clinical Neuropsychology*, 2011.
- Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch, I., and Florence, P. PaLM-E: An embodied multimodal language model, March 2023.
- Duan, Y., Andrychowicz, M., Stadie, B. C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning, 2017.
- Ebert, F., Finn, C., Dasari, S., Xie, A., Lee, A. X., and Levine, S. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *ArXiv*, abs/1812.00568, 2018.
- Ebert, F., Yang, Y., Schmeckpeper, K., Bucher, B., Georgakis, G., Daniilidis, K., Finn, C., and Levine, S. Bridge data: Boosting generalization of robotic skills with cross-domain datasets, 2021.
- Elgeneidy, K., Lightbody, P., Pearson, S., and Neumann, G. Characterising 3d-printed soft fin ray robotic fingers with layer jamming capability for delicate grasping. In *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 143–148, 2019. doi: 10.1109/ROBOSOFT.2019.8722715.

- Elsner, J., Reinerth, G., Figueredo, L., Naceri, A., Walter, U., and Haddadin, S. Parti-a haptic virtual reality control station for model-mediated robotic applications. *Frontiers in Virtual Reality*, 3:925794, 2022.
- Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. Rvs: What is essential for offline rl via supervised learning? *ArXiv*, abs/2112.10751, 2021.
- Escontrela, A., Peng, X. B., Yu, W., Zhang, T., Iscen, A., Goldberg, K., and Abbeel, P. Adversarial motion priors make good substitutes for complex reward functions, 2022.
- Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.
- Eysenbach, B., Levine, S., and Salakhutdinov, R. Replacing rewards with examples: Example-based policy search via recursive classification. In *NeurIPS*, 2021.
- Fang, H., Fang, H.-S., Wang, Y., Ren, J., Chen, J., Zhang, R., Wang, W., and Lu, C. Low-cost exoskeletons for learning whole-arm manipulation in the wild. *arXiv preprint arXiv:2309.14975*, 2023a.
- Fang, H.-S., Fang, H., Tang, Z., Liu, J., Wang, J., Zhu, H., and Lu, C. Rh20t: A robotic dataset for learning diverse skills in one-shot, 2023b.
- Fang, K., Yin, P., Nair, A., Walke, H., Yan, G., and Levine, S. Generalization with lossy affordances: Leveraging broad offline data for learning visuomotor tasks, 2023c.
- Fikes, R. E. and Nilsson, N. J. Strips: A new approach to the application of theorem proving to problem solving. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, IJCAI'71, pp. 608–620, San Francisco, CA, USA, 1971. Morgan Kaufmann Publishers Inc.
- Finn, C. and Levine, S. Deep visual foresight for planning robot motion. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 2786–2793. IEEE, 2017.
- Finn, C., Goodfellow, I., and Levine, S. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pp. 64–72, 2016a.
- Finn, C., Levine, S., and Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. *ArXiv*, abs/1603.00448, 2016b.
- Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017a.
- Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. One-shot visual imitation learning via meta-learning. In *CoRL*, 2017b.

- Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. *ArXiv*, abs/1710.11248, 2018a.
- Fu, J., Singh, A., Ghosh, D., Yang, L., and Levine, S. Variational inverse control with events: A general framework for data-driven reward definition. In *NeurIPS*, 2018b.
- Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning, 2020.
- Fu, Z., Zhao, T. Z., and Finn, C. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.
- Fujimoto, S., Meger, D., and Precup, D. Off-Policy Deep Reinforcement Learning without Exploration. *CoRR*, abs/1812.02900, 2018a.
- Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018b.
- Gal, Y., McAllister, R., and Rasmussen, C. E. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, 2016.
- Garrett, C. R., Lozano-Pérez, T., and Kaelbling, L. P. FFRob: Leveraging symbolic planning for efficient task and motion planning. *The International Journal of Robotics Research*, 37(1):104–136, nov 2017. doi: 10.1177/0278364917739114. URL <https://doi.org/10.1177/0278364917739114>.
- Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., and Lozano-Pérez, T. Integrated task and motion planning, 2020a.
- Garrett, C. R., Lozano-Pérez, T., and Kaelbling, L. P. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning, 2020b.
- Gealy, D. V., McKinley, S., Yi, B., Wu, P., Downey, P. R., Balke, G., Zhao, A., Guo, M., Thomasson, R., Sinclair, A., Cuellar, P., McCarthy, Z., and Abbeel, P. Quasi-direct drive for low-cost compliant robotic manipulation, 2019.
- Ghallab, M., Howe, A., Knoblock, C., Mcdermott, D., Ram, A., Veloso, M., Weld, D., and Wilkins, D. PDDL—The Planning Domain Definition Language, 1998. URL <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.212>.
- Ghallab, M., Nau, D., and Traverso, P. *Automated Planning: Theory and Practice*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, Amsterdam, 2004. ISBN 978-1-55860-856-6. URL <http://www.sciencedirect.com/science/book/9781558608566>.

- Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.
- Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative adversarial nets. In *NIPS*, 2014.
- Gulcehre, C., Wang, Z., Novikov, A., Paine, T. L., Colmenarejo, S. G., Zolna, K., Agarwal, R., Merel, J., Mankowitz, D. J., Paduraru, C., Dulac-Arnold, G., Li, J. Z., Norouzi, M., Hoffman, M. W., Nachum, O., Tucker, G., Heess, N. M. O., and de Freitas, N. Rl unplugged: Benchmarks for offline reinforcement learning. *ArXiv*, abs/2006.13888, 2020.
- Guo, J., Liu, C., and Poignet, P. A scaled bilateral teleoperation system for robotic-assisted surgery with time delay. *Journal of Intelligent & Robotic Systems*, 95, 07 2019a. doi: 10.1007/s10846-018-0918-1.
- Guo, M., Wu, P., Yi, B., Gealy, D., McKinley, S., and Abbeel, P. Blue gripper: A robust, low-cost, and force-controlled robot hand. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pp. 1505–1510, 2019b. doi: 10.1109/COASE.2019.8843134.
- Gupta, A., Kumar, V., Lynch, C., Levine, S., and Hausman, K. Relay policy learning: Solving long horizon tasks via imitation and reinforcement learning. *Conference on Robot Learning (CoRL)*, 2019.
- Ha, H. and Song, S. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. *Conference on Robot Learning*, 2021.
- Ha, H., Florence, P., and Song, S. Scaling up and distilling down: Language-guided robot skill acquisition, July 2023.
- Ha, S., Xu, P., Tan, Z., Levine, S., and Tan, J. Learning to walk in the real world with minimal human effort, 2020a.
- Ha, S., Xu, P., Tan, Z., Levine, S., and Tan, J. Learning to walk in the real world with minimal human effort. *arXiv preprint arXiv:2002.08550*, 2020b.
- Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018a.
- Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.
- Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

- Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565, 2019b.
- Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.
- Hafner, D., Lee, K.-H., Fischer, I., and Abbeel, P. Deep hierarchical planning from pixels, 2022.
- Hailat, Z. and Chen, X.-W. Teacher/student deep semi-supervised learning for training with noisy labels. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018.
- Haldar, S., Pari, J., Rai, A., and Pinto, L. Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023.
- Handa, A., Van Wyk, K., Yang, W., Liang, J., Chao, Y.-W., Wan, Q., Birchfield, S., Ratliff, N., and Fox, D. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9164–9170. IEEE, 2020.
- Hannaford, B. A design framework for teleoperators with kinesthetic feedback. *IEEE Transactions on Robotics and Automation*, 5(4):426–434, 1989. doi: 10.1109/70.88057.
- Hansen, N., Lin, Y., Su, H., Wang, X., Kumar, V., and Rajeswaran, A. Modem: Accelerating visual model-based reinforcement learning with demonstrations. *arXiv preprint*, 2022a.
- Hansen, N., Wang, X., and Su, H. Temporal difference learning for model predictive control. In *ICML*, 2022b.
- He, K., Chen, X., Xie, S., Li, Y., Doll’ar, P., and Girshick, R. B. Masked autoencoders are scalable vision learners. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15979–15988, 2021.
- Henaff, M., Canziani, A., and LeCun, Y. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. *arXiv preprint arXiv:1901.02705*, 2019.
- Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. *ArXiv*, abs/1808.06670, 2019.

- Ho, J. and Ermon, S. Generative adversarial imitation learning. In *NIPS*, 2016.
- Holmberg, R. and Khatib, O. Development and control of a holonomic mobile robot for mobile manipulation tasks. *The International Journal of Robotics Research*, 19(11):1066–1074, 2000. doi: 10.1177/02783640022067977. URL <https://doi.org/10.1177/02783640022067977>.
- Hoque, R., Balakrishna, A., Novoseller, E., Wilcox, A., Brown, D. S., and Goldberg, K. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning, 2021.
- Hsu, K., Kim, M. J., Rafailov, R., Wu, J., and Finn, C. Vision-based manipulators need to also see from their hands, 2022.
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=nZeVKeeFYf9>.
- Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, January 2022.
- Huang, W., Xia, F., Shah, D., Driess, D., Zeng, A., Lu, Y., Florence, P., Mordatch, I., Levine, S., Hausman, K., and Ichter, B. Grounded decoding: Guiding text generation with grounded models for robot control, March 2023.
- Hulin, T., Hertkorn, K., Kremer, P., Schätzle, S., Artigas, J., Sagardia, M., Zacharias, F., and Preusche, C. The dlr bimanual haptic device with optimized workspace. In *2011 IEEE International Conference on Robotics and Automation*, pp. 3441–3442. IEEE, 2011.
- Intel. Realsense d405. <https://www.intelrealsense.com/depth-camera-d405/>, 2024.
- Irpan, A., Harris, C., Ibarz, J., Rao, K., Khansari, M., and Levine, S. Rl-cyclegan: Improving deep-rl robotics with simulation-to-real. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, 2020.
- Ishiguro, Y., Makabe, T., Nagamatsu, Y., Kojio, Y., Kojima, K., Sugai, F., Kakiuchi, Y., Okada, K., and Inaba, M. Bilateral humanoid teleoperation system using whole-body exoskeleton cockpit tablis. *IEEE Robotics and Automation Letters*, 5(4):6419–6426, 2020.
- James, S. and Davison, A. J. Q-attention: Enabling efficient learning for vision-based robotic manipulation, 2021.
- James, S., Bloesch, M., and Davison, A. J. Task-embedded control networks for few-shot imitation learning. *ArXiv*, abs/1810.03237, 2018.
- James, S., Wada, K., Laidlow, T., and Davison, A. J. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation, 2021.

- Jang, E., Irpan, A., Khansari, M., Kappler, D., Ebert, F., Lynch, C., Levine, S., and Finn, C. BC-z: Zero-shot task generalization with robotic imitation learning. In *5th Annual Conference on Robot Learning*, 2021.
- Jang, E., Irpan, A., Khansari, M., Kappler, D., Ebert, F., Lynch, C., Levine, S., and Finn, C. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pp. 991–1002. PMLR, 2022.
- Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *ArXiv*, abs/1906.08253, 2019.
- Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021.
- Jiang, Y., Zhang, S., Khandelwal, P., and Stone, P. Task planning in robotics: an empirical comparison of pddl-based and asp-based systems, 2019.
- Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., and Fan, L. Vima: General robot manipulation with multimodal prompts, 2023.
- Jiang, Z., Zhang, T., Janner, M., Li, Y., Rocktäschel, T., Grefenstette, E., and Tian, Y. Efficient planning in a compact latent action space. *arXiv preprint arXiv:2208.10291*, 2022.
- Jing, L. and Tian, Y. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:4037–4058, 2019.
- Jo, I., Park, Y., and Bae, J. A teleoperation system with an exoskeleton interface. In *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1649–1654. IEEE, 2013.
- Jolliffe, I. T. and Cadima, J. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374, 2016.
- Kaelbling, L. P. and Lozano-Pérez, T. Integrated task and motion planning in belief space. *Int. J. Rob. Res.*, 32(9–10):1194–1227, aug 2013. ISSN 0278-3649. doi: 10.1177/0278364913484072. URL <https://doi.org/10.1177/0278364913484072>.
- Kaelbling, L. P. and Lozano-Pérez, T. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pp. 1470–1477, 2011. doi: 10.1109/ICRA.2011.5980391.
- Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

- Kalashnikov, D., Varley, J., Chebotar, Y., Swanson, B., Jonschkowski, R., Finn, C., Levine, S., and Hausman, K. Mt-opt: Continuous multi-task robotic reinforcement learning at scale, 2021.
- Katyal, K. D., Brown, C. Y., Hechtman, S. A., Para, M. P., McGee, T. G., Wolfe, K. C., Murphy, R. J., Kutzer, M. D., Tunstel, E. W., McLoughlin, M. P., et al. Approaches to robotic teleoperation in a disaster scenario: From supervised autonomy to direct control. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1874–1881. IEEE, 2014.
- Katz, B. A low cost modular actuator for dynamic robots. In <https://api.semanticscholar.org/CorpusID:69840472>, 2018a.
- Katz, B. A low cost modular actuator for dynamic robots. *PhD Thesis*, 2018b.
- Ke, T.-W., Gkanatsios, N., and Fragkiadaki, K. 3d diffuser actor: Policy diffusion with 3d scene representations, 2024.
- Kelly, M., Sidrane, C., Driggs-Campbell, K., and Kochenderfer, M. J. Hg-dagger: Interactive imitation learning with human experts, 2019.
- Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. MOREL : Model-Based Offline Reinforcement Learning. In *NeurIPS*, 2020.
- Kim, S.-K., Hong, S., and Kim, D. A walking motion imitation framework of a humanoid robot by human walking recognition from imu motion data. *9th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS09*, pp. 343 – 348, 01 2010. doi: 10.1109/ICHR.2009.5379552.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., and Girshick, R. B. Segment anything. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3992–4003, 2023. URL <https://api.semanticscholar.org/CorpusID:257952310>.
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13): 3521–3526, 2017.

- Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. *ArXiv*, 2021.
- Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-Learning for Offline Reinforcement Learning. *ArXiv*, abs/2006.04779, 2020.
- Kumar, A., Fu, Z., Pathak, D., and Malik, J. Rma: Rapid motor adaptation for legged robots, 2021.
- Lab, D. H. "phantom omni" - 6 dof master device, 2023. URL <https://delfthapticslab.nl/device/phantom-omni/>.
- Laghi, M., Maimeri, M., Marchand, M., Leparoux, C., Catalano, M., Ajoudani, A., and Bicchi, A. Shared-autonomy control for intuitive bimanual tele-manipulation. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pp. 1–9. IEEE, 2018.
- Lai, X., Tian, Z., Chen, Y., Li, Y., Yuan, Y., Liu, S., and Jia, J. Lisa: Reasoning segmentation via large language model. *arXiv preprint arXiv:2308.00692*, 2023.
- Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. In *Reinforcement learning*, pp. 45–73. Springer, 2012.
- Laskey, M., Lee, J., Chuck, C., Gealy, D., Hsieh, W., Pokorny, F. T., Dragan, A. D., and Goldberg, K. Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 827–834, 2016. doi: 10.1109/COASE.2016.7743488.
- Laskey, M., Lee, J., Fox, R., Dragan, A., and Goldberg, K. Dart: Noise injection for robust imitation learning, 2017.
- Laskin, M., Yarats, D., Liu, H., Lee, K., Zhan, A., Lu, K., Cang, C., Pinto, L., and Abbeel, P. Urlb: Unsupervised reinforcement learning benchmark. *arXiv preprint arXiv:2110.15191*, 2021.
- Lawrence, D. Stability and transparency in bilateral teleoperation. *IEEE Transactions on Robotics and Automation*, 9(5):624–637, 1993. doi: 10.1109/70.258054.
- Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., and Hutter, M. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47), oct 2020. doi: 10.1126/scirobotics.abc5986.
- Lenz, C. and Behnke, S. Bimanual telemanipulation with force and haptic feedback and predictive limit avoidance. *CoRR*, abs/2109.13382, 2021.
- Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

- Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., and Quillen, D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.
- Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.
- Li, A., Wu, P., and Kennedy, M. Replay Overshooting: Learning stochastic latent dynamics with the extended kalman filter. In *2021 International Conference on Robotics and Automation (ICRA)*, 2021.
- Li, J., Li, D., Savarese, S., and Hoi, S. C. H. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning*, 2023a.
- Li, Q., Peng, Z., and Zhou, B. Efficient learning of safe driving policy via human-ai copilot optimization, 2022.
- Li, X., Liu, M., Zhang, H., Yu, C., Xu, J., Wu, H., Cheang, C., Jing, Y., Zhang, W., Liu, H., Li, H., and Kong, T. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023b.
- Liang, J., Mahler, J., Laskey, M., Li, P., and Goldberg, K. Using dvrk teleoperation to facilitate deep learning of automation tasks for an industrial robot. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pp. 1–8, 2017. doi: 10.1109/COASE.2017.8256067.
- Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. Code as policies: Language model programs for embodied control, September 2022.
- Lichiardopol, S. A survey on teleoperation, 2007.
- Lifschitz, V. What is answer set programming? In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3, AAAI’08*, pp. 1594–1597. AAAI Press, 2008. ISBN 9781577353683.
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- Lin, B. Y., Huang, C., Liu, Q., Gu, W., Sommerer, S., and Ren, X. On grounded planning for embodied tasks with language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):13192–13200, June 2023. ISSN 2159-5399. doi: 10.1609/aaai.v37i11.26549. URL <http://dx.doi.org/10.1609/aaai.v37i11.26549>.

- Lin, X., So, J., Mahalingam, S., Liu, F., and Abbeel, P. Spawnnet: Learning generalizable visuomotor skills from pre-trained network, 05 2024.
- Liu, F., Liu, H., Grover, A., and Abbeel, P. Masked autoencoding for scalable and generalizable decision making, 2023a.
- Liu, F., Fang, K., Abbeel, P., and Levine, S. Moka: Open-vocabulary robotic manipulation through mark-based visual prompting, 2024a.
- Liu, H. and Abbeel, P. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pp. 6736–6747. PMLR, 2021.
- Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual instruction tuning, 2023b.
- Liu, H., Nasiriany, S., Zhang, L., Bao, Z., and Zhu, Y. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. *RSS*, 2023c.
- Liu, P., Orru, Y., Vakil, J., Paxton, C., Shafiullah, N. M. M., and Pinto, L. Ok-robot: What really matters in integrating open-knowledge models for robotics, 2024b.
- Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023d.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, 2019.
- Liu, Y., Swaminathan, A., Agarwal, A., and Brunskill, E. Provably good batch off-policy reinforcement learning without great exploration. In *Neural Information Processing Systems*, 2020.
- Liu, Z., Bahety, A., and Song, S. Reflect: Summarizing robot experiences for failure explanation and correction. *arXiv preprint arXiv:2306.15724*, 2023e.
- Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2019.
- Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., and Zhang, Y. An empirical study of catastrophic forgetting in large language models during continual fine-tuning, 2023.
- Lynch, C. and Sermanet, P. Language conditioned imitation learning over unstructured data. *Robotics: Science and Systems*, 2021.
- Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. Learning latent plans from play. *arXiv preprint arXiv:1903.01973*, 2019.

- Lynch, C., Wahid, A., Tompson, J., Ding, T., Betker, J., Baruch, R., Armstrong, T., and Florence, P. Interactive language: Talking to robots in real time, October 2022.
- Lynch, K. M. and Park, F. C. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, USA, 1st edition, 2017. ISBN 1107156300.
- Ma, R. and Dollar, A. Yale openhand project: Optimizing open-source hand designs for ease of fabrication and adoption. *IEEE Robotics and Automation Magazine*, 24(1):32–40, 2017. doi: 10.1109/MRA.2016.2639034.
- Majumdar, A., Yadav, K., Arnaud, S., Ma, Y. J., Chen, C., Silwal, S., Jain, A., Berges, V.-P., Abbeel, P., Malik, J., Batra, D., Lin, Y., Maksymets, O., Rajeswaran, A., and Meier, F. Where are we in the search for an artificial visual cortex for embodied intelligence?, 2023.
- Mandi, Z., Liu, F., Lee, K., and Abbeel, P. Towards more generalizable one-shot visual imitation learning, 2021.
- Mandlekar, A., Xu, D., Martín-Martín, R., Zhu, Y., Fei-Fei, L., and Savarese, S. Human-in-the-loop imitation learning using remote teleoperation. *arXiv preprint arXiv:2012.06733*, 2020.
- Mandlekar, A., Xu, D., Wong, J., Nasiriany, S., Wang, C., Kulkarni, R., Fei-Fei, L., Savarese, S., Zhu, Y., and Martín-Martín, R. What matters in learning from offline human demonstrations for robot manipulation. *ArXiv*, abs/2108.03298, 2021.
- Mansouri, M., Pecora, F., and Schüller, P. Combining task and motion planning: Challenges and guidelines. *Frontiers in Robotics and AI*, 8, 2021. ISSN 2296-9144. doi: 10.3389/frobt.2021.637888. URL <https://www.frontiersin.org/articles/10.3389/frobt.2021.637888>.
- Mees, O., Hermann, L., Rosete-Beas, E., and Burgard, W. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks, 2022.
- Meta. Meta quest 2. <https://www.meta.com/quest/products/quest-2/>, 2024.
- Miki, T., Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., and Hutter, M. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62), jan 2022. doi: 10.1126/scirobotics.abk2822.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
- Mohsen, M., Mohamed, A. M., Ahmed, S., and Ibrahim, K. Bilateral control of a 2-dof teleoperated manipulator using udp scheme. *Ain Shams Engineering Journal*, 14(9):102065, 2023. ISSN 2090-4479.

- Myers, V., He, A., Fang, K., Walke, H., Hansen-Estruch, P., Cheng, C.-A., Jalobeanu, M., Kolobov, A., Dragan, A., and Levine, S. Goal representations for instruction following: A semi-supervised language interface to control, 2023.
- Nagabandi, A., Yang, G., Asmar, T., Pandya, R., Kahn, G., Levine, S., and Fearing, R. S. Learning image-conditioned dynamics models for control of under-actuated legged millirobots, 2017.
- Nagabandi, A., Konoglie, K., Levine, S., and Kumar, V. Deep dynamics models for learning dexterous manipulation, 2019.
- Naik, D. K. and Mammone, R. Meta-neural networks that learn by learning. In *International Joint Conference on Neural Networks (IJCNN)*, 1992.
- Nair, S., Rajeswaran, A., Kumar, V., Finn, C., and Gupta, A. R3m: A universal visual representation for robot manipulation, 2022.
- Okada, M. and Taniguchi, T. Dreaming: Model-based reinforcement learning by latent imagination without reconstruction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4209–4215. IEEE, 2021.
- Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- OpenAI. Gpt-4 technical report, 2023.
- OpenAI, Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. Learning dexterous in-hand manipulation, 2018.
- Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. ISSN 0893-6080.
- Parisi, S., Rajeswaran, A., Purushwalkam, S., and Gupta, A. K. The unsurprising effectiveness of pre-trained vision models for control. In *ICML*, 2022.
- Paster, K., McKinney, L. E., McIlraith, S. A., and Ba, J. Blast: Latent dynamics models from bootstrapping. In *Deep RL Workshop NeurIPS 2021*, 2021.
- Pastor, P., Hoffmann, H., Asfour, T., and Schaal, S. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pp. 763–768, 2009. doi: 10.1109/ROBOT.2009.5152385.
- Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

- Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, May 2018. doi: 10.1109/ICRA.2018.8460528.
- Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- Pinto, L. and Gupta, A. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours, 2015.
- Plaku, E. and Hager, G. D. Sampling-based motion and symbolic action planning with geometric and differential constraints. In *2010 IEEE International Conference on Robotics and Automation*, pp. 5002–5008, 2010. doi: 10.1109/ROBOT.2010.5509563.
- Pomerleau, D. A. Alvin: An autonomous land vehicle in a neural network. In Touretzky, D. (ed.), *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.
- Powell, W. B. *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics)*. Wiley-Interscience, USA, 2007. ISBN 0470171553.
- Pratt, G. and Williamson, M. Series elastic actuators. In *2011 IEEE International Conference on Robotics and Automation*, 1995.
- Pugin, F., Bucher, P., and Morel, P. History of robotic surgery : From aesop® and zeus® to da vinci®. *Journal of Visceral Surgery*, 148(5, Supplement):e3–e8, 2011. ISSN 1878-7886. Robotic surgery.
- Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., and Torralba, A. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, 2018.
- Qin, Y., Wu, Y.-H., Liu, S., Jiang, H., Yang, R., Fu, Y., and Wang, X. Dexmv: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision*, pp. 570–587. Springer, 2022.
- Qin, Y., Yang, W., Huang, B., Van Wyk, K., Su, H., Wang, X., Chao, Y.-W., and Fox, D. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *RSS*, 2023.
- Quigley, M., Asbeck, A., and Ng, A. A low-cost compliant 7-dof robotic manipulator. In *2011 IEEE International Conference on Robotics and Automation*, 2011.

- Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision, 2021.
- Radosavovic, I., Wang, X., Pinto, L., and Malik, J. State-only imitation learning for dexterous manipulation. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7865–7871, 2021.
- Rafailov, R., Yu, T., Rajeswaran, A., and Finn, C. Visual adversarial imitation learning using variational models. In *NeurIPS*, 2021.
- Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.
- Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *NeurIPS*, 2019.
- Rajeswaran, A., Mordatch, I., and Kumar, V. A Game Theoretic Framework for Model-Based Reinforcement Learning. In *ICML*, 2020.
- Rakita, D., Mutlu, B., and Gleicher, M. A motion retargeting method for effective mimicry-based teleoperation of robot arms. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 361–370, 2017.
- Rana, K., Haviland, J., Garg, S., Abou-Chakra, J., Reid, I., and Suenderhauf, N. Sayplan: Grounding large language models using 3d scene graphs for scalable task planning. *arXiv preprint arXiv:2307.06135*, 2023.
- Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A. D., Heess, N. M. O., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. A generalist agent. *ArXiv*, abs/2205.06175, 2022.
- Ren, A. Z., Dixit, A., Bodrova, A., Singh, S., Tu, S., Brown, N., Xu, P., Takayama, L., Xia, F., Varley, J., Xu, Z., Sadigh, D., Zeng, A., and Majumdar, A. Robots that ask for help: Uncertainty alignment for large language model planners, July 2023.
- Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. Meta-learning for semi-supervised few-shot classification. *ArXiv*, abs/1803.00676, 2018.
- Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.
- ROBOTIS, I. Dynamixel xl, 2023. URL <https://www.robotis.us/xl/>.

- Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.
- Ross, S., Melik-Barkhudarov, N., Shankar, K. S., Wendel, A., Dey, D., Bagnell, J. A., and Hebert, M. Learning monocular reactive uav control in cluttered natural environments, 2012.
- Rudin, N., Hoeller, D., Reist, P., and Hutter, M. Learning to walk in minutes using massively parallel deep reinforcement learning, 2021.
- Rusu, A. A., Vecerik, M., Rothörl, T., Heess, N., Pascanu, R., and Hadsell, R. Sim-to-real robot learning from pixels with progressive nets, 2016.
- Schaal, S. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, 1999. ISSN 1364-6613. doi: [https://doi.org/10.1016/S1364-6613\(99\)01327-3](https://doi.org/10.1016/S1364-6613(99)01327-3).
- Schmidhuber, J. Evolutionary principles in self-referential learning. *Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1987.
- Schmidhuber, J. Reinforcement learning upside down: Don’t predict rewards – just map them to actions, 2019.
- Schoettler, G., Nair, A., Luo, J., Bahl, S., Ojea, J. A., Solowjow, E., and Levine, S. Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards, 2019.
- Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.
- Schwarz, M., Lenz, C., Rochow, A., Schreiber, M., and Behnke, S. Nimbro avatar: Interactive immersive telepresence with force-feedback telemanipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5312–5319. IEEE, 2021.
- Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pp. 8583–8592. PMLR, 2020.
- Seo, Y., Hafner, D., Liu, H., Liu, F., James, S., Lee, K., and Abbeel, P. Masked world models for visual control. *ArXiv*, abs/2206.14244, 2022.

- Shafiullah, N. M. M., Cui, Z. J., Altanzaya, A., and Pinto, L. Behavior transformers: Cloning k modes with one stone. *ArXiv*, abs/2206.11251, 2022.
- Shafiullah, N. M. M., Rai, A., Etukuru, H., Liu, Y., Misra, I., Chintala, S., and Pinto, L. On bringing robots home. *arXiv preprint arXiv:2311.16098*, 2023.
- Shah, D. and Levine, S. Viking: Vision-based kilometer-scale navigation with geographic hints, 2022.
- Shan, X. and Birglen, L. Bistable Stopper Design and Force Prediction for Precision and Power Grasps of Soft Robotic Fingers for Industrial Manipulation. *Journal of Mechanical Design*, 146(4):045001, 11 2023. ISSN 1050-0472. doi: 10.1115/1.4063763.
- Shaw, K., Agarwal, A., and Pathak, D. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *Robotics: Science and Systems (RSS)*, 2023.
- Shridhar, M., Manuelli, L., and Fox, D. Cliport: What and where pathways for robotic manipulation, 2021.
- Shridhar, M., Manuelli, L., and Fox, D. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pp. 785–799. PMLR, 2023.
- Siciliano, B. and Khatib, O. *Springer Handbook of Robotics*. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 354023957X.
- Siekman, J., Green, K., Warila, J., Fern, A., and Hurst, J. Blind bipedal stair traversal via sim-to-real reinforcement learning, 2021.
- Siméon, T., Laumond, J.-P., Cortés, J., and Sahbani, A. Manipulation planning with probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7-8):729–746, 2004. doi: 10.1177/0278364904045471. URL <https://doi.org/10.1177/0278364904045471>.
- Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530. IEEE, 2023.
- Sivakumar, A., Shaw, K., and Pathak, D. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube, 2022.
- Skreta, M., Yoshikawa, N., Arellano-Rubach, S., Ji, Z., Kristensen, L. B., Darvish, K., Aspuru-Guzik, A., Shkurti, F., and Garg, A. Errors are useful prompts: Instruction guided task programming with verifier-assisted iterative prompting. *arXiv preprint arXiv:2303.14100*, 2023.
- Smith, L., Dhawan, N., Zhang, M., Abbeel, P., and Levine, S. Avid: Learning multi-stage tasks via pixel-level translation of human videos. *RSS*, 2019.

- Smith, L., Kew, J. C., Peng, X. B., Ha, S., Tan, J., and Levine, S. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world, 2021.
- Smith, L., Kostrikov, I., and Levine, S. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning, 2022.
- Song, S., Zeng, A., Lee, J., and Funkhouser, T. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3): 4978–4985, 2020.
- Spencer, J., Choudhury, S., Barnes, M., Schmittle, M., Chiang, M., Ramadge, P. J., and Srinivasa, S. S. Learning from interventions: Human-robot interaction as both explicit and implicit feedback. *Robotics: Science and Systems XVI*, 2020.
- Spencer, J., Choudhury, S., Venkatraman, A., Ziebart, B., and Bagnell, J. A. Feedback in imitation learning: The three regimes of covariate shift. *arXiv preprint arXiv:2102.02872*, 2021.
- Srivastava, R. K., Shyam, P., Mutz, F., Jaśkowski, W., and Schmidhuber, J. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.
- Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.
- Tachi, S., Komoriya, K., Sawada, K., Nishiyama, T., Itoko, T., Kobayashi, M., and Inoue, K. Telexistence cockpit for humanoid robot control, 2003.
- Thrun, S. and Pratt, L. *Learning to learn*. Springer Science & Business Media, 1998.
- Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.
- Toedtheide, A., Chen, X., Sadeghian, H., Naceri, A., and Haddadin, S. A force-sensitive exoskeleton for teleoperation: An application in elderly care robotics. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12624–12630. IEEE, 2023.
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023.
- Triguero, I., García, S., and Herrera, F. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems*, 42:245–284, 2013.

- Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., and Tassa, Y. *dm_control: Software and tasks for continuous control. Software Impacts*, 6:100022, 2020. ISSN 2665-9638.
- Tzeng, E., Devin, C., Hoffman, J., Finn, C., Abbeel, P., Levine, S., Saenko, K., and Darrell, T. Adapting deep visuomotor representations with weak pairwise constraints, 2015.
- van Engelen, J. E. and Hoos, H. H. A survey on semi-supervised learning. *Machine Learning*, 109:373–440, 2019.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.
- Vemprala, S., Bonatti, R., Bucker, A. F. C., and Kapoor, A. Chatgpt for robotics: Design principles and model abilities. *ArXiv*, abs/2306.17582, 2023. URL <https://api.semanticscholar.org/CorpusID:259141622>.
- Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *ICML '08*, 2008.
- Vinyals, O., Blundell, C., Lillicrap, T. P., Kavukcuoglu, K., and Wierstra, D. Matching networks for one shot learning. In *NIPS*, 2016.
- Vive. Vive vr products, 2023.
- Walke, H., Black, K., Lee, A., Kim, M. J., Du, M., Zheng, C., Zhao, T., Hansen-Estruch, P., Vuong, Q., He, A., et al. Bridgedata v2: A dataset for robot learning at scale. *arXiv preprint arXiv:2308.12952*, 2023.
- Wang, A., Ramos, J., Mayo, J., Ubellacker, W., Cheung, J., and Kim, S. The hermes humanoid system: A platform for full-body teleoperation with balance feedback. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 730–737, 2015. doi: 10.1109/HUMANOIDS.2015.7363451.
- Wang, C., Fan, L., Sun, J., Zhang, R., Fei-Fei, L., Xu, D., Zhu, Y., and Anandkumar, A. Mimicplay: Long-horizon imitation learning by watching human play. *arXiv preprint arXiv:2302.12422*, 2023a.
- Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models, 2023b.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models, 2023.
- Wen, C., Lin, X., So, J., Chen, K., Dou, Q., Gao, Y., and Abbeel, P. Any-point trajectory modeling for policy learning. *arXiv preprint arXiv:2401.00025*, 2023.

- Whitney, J., Glisson, M., Brockmeyer, E., and Hodgins, J. A low-friction passive fluid transmission and fluid-tendon soft actuator. *IEEE International Conference on Intelligent Robots and Systems*, pp. 2801–2808, 10 2014. doi: 10.1109/IROS.2014.6942946.
- Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.
- Wu, J., Antonova, R., Kan, A., Lepert, M., Zeng, A., Song, S., Bohg, J., Rusinkiewicz, S., and Funkhouser, T. TidyBot: Personalized robot assistance with large language models, May 2023a.
- Wu, J., Antonova, R., Kan, A., Lepert, M., Zeng, A., Song, S., Bohg, J., Rusinkiewicz, S., and Funkhouser, T. Tidybot: Personalized robot assistance with large language models. *Autonomous Robots*, 2023b.
- Wu, P., Escontrela, A., Hafner, D., Goldberg, K., and Abbeel, P. Daydreamer: World models for physical robot learning. *ArXiv*, abs/2206.14176, 2022.
- Wu, P., Li, B., Abbeel, P., and Malik, J. Interactive task planning with language models, 2023c.
- Wu, P., Majumdar, A., Stone, K., Lin, Y., Mordatch, I., Abbeel, P., and Rajeswaran, A. Masked trajectory models for prediction, representation, and control. In *International Conference on Machine Learning*, 2023d.
- Wu, P., Shentu, Y., Yi, Z., Lin, X., and Abbeel, P. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators, 2023e.
- Wu, P., Hakhamaneshi, K., Du, Y., Mordatch, I., Rajeswaran, A., and Abbeel, P. Semi-supervised one shot imitation learning. *Reinforcement Learning Journal*, 5:2284–2297, 2024a.
- Wu, P., Shentu, Y., Liao, Q., Jin, D., Guo, M., Sreenath, K., Lin, X., and Abbeel, P. Robocopilot: Human-in-the-loop interactive imitation learning for robot manipulation, 2024b.
- Wu, Y., Balatti, P., Lorenzini, M., Zhao, F., Kim, W., and Ajoudani, A. A teleoperation interface for loco-manipulation control of mobile collaborative robotic assistant. *IEEE Robotics and Automation Letters*, 4(4):3593–3600, 2019. doi: 10.1109/LRA.2019.2928757.
- Wyrobek, K. A., Berger, E. H., der Loos, H. M. V., and Salisbury, J. K. Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. *2008 IEEE International Conference on Robotics and Automation*, pp. 2165–2170, 2008.
- Xiao, T., Radosavovic, I., Darrell, T., and Malik, J. Masked visual pre-training for motor control, 2022.

- Xie, A., Ebert, F., Levine, S., and Finn, C. Improvisation through physical understanding: Using novel objects as tools with visual foresight. *arXiv preprint arXiv:1904.05538*, 2019.
- Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10687–10698, 2020.
- Xiong, H., Mendonca, R., Shaw, K., and Pathak, D. Adaptive mobile manipulation for articulated objects in the open world. *arXiv preprint arXiv:2401.14403*, 2024.
- Yang, J., Zhang, H., Li, F., Zou, X., Li, C., and Gao, J. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v, 2023.
- Yang, M. and Nachum, O. Representation matters: Offline pretraining for sequential decision making, 2021.
- Yang, M., Du, Y., Ghasemipour, K., Tompson, J., Kaelbling, L., Schuurmans, D., and Abbeel, P. Learning interactive real-world simulators, 2024.
- Yang, S.-B. and Yu, T.-l. Pseudo-representation labeling semi-supervised learning, 2020.
- Yang, T.-Y., Zhang, T., Luu, L., Ha, S., Tan, J., and Yu, W. Safe reinforcement learning for legged locomotion, 2022a.
- Yang, Y., Caluwaerts, K., Iscen, A., Zhang, T., Tan, J., and Sindhvani, V. Data efficient reinforcement learning for legged robots, 2019.
- Yang, Y., Zhang, T., Coumans, E., Tan, J., and Boots, B. Fast and efficient locomotion via learned gait transitions. In *Conference on Robot Learning*, pp. 773–783. PMLR, 2022b.
- Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021a.
- Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Reinforcement learning with prototypical representations. *ArXiv*, 2021b.
- Yarats, D., Brandfonbrener, D., Liu, H., Laskin, M., Abbeel, P., Lazaric, A., and Pinto, L. Don’t change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.
- Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, 1995.
- Yide Shentu, Philipp Wu, A. R. and Abbeel, P. From llms to actions: Latent codes as bridges in hierarchical robot control, 2024.

- Young, S., Gandhi, D., Tulsiani, S., Gupta, A., Abbeel, P., and Pinto, L. Visual imitation made easy. *CoRL*, 2020.
- Yu, L., Yu, T., Finn, C., and Ermon, S. Meta-inverse reinforcement learning with probabilistic context variables. *ArXiv*, abs/1909.09314, 2019.
- Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.
- Yu, T., Kumar, A., Chebotar, Y., Hausman, K., Finn, C., and Levine, S. How to leverage unlabeled data in offline reinforcement learning. *ArXiv*, abs/2202.01741, 2022.
- Zeng, A., Attarian, M., Ichter, B., Choromanski, K., Wong, A., Welker, S., Tombari, F., Purohit, A., Ryoo, M., Sindhvani, V., Lee, J., Vanhoucke, V., and Florence, P. Socratic models: Composing zero-shot multimodal reasoning with language, April 2022.
- Zeng, F., Gan, W., Wang, Y., Liu, N., and Yu, P. S. Large language models for robotics: A survey. *ArXiv*, abs/2311.07226, 2023. URL <https://api.semanticscholar.org/CorpusID:265149884>.
- ZeroMQ. Zeromq. <https://zeromq.org/>, 2024.
- Zhang, J. and Cho, K. Query-efficient imitation learning for end-to-end autonomous driving, 2016. URL <https://arxiv.org/abs/1605.06450>.
- Zhang, M., Vikram, S., Smith, L., Abbeel, P., Johnson, M., and Levine, S. Solar: deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning*, 2019.
- Zhang, T., McCarthy, Z., Jow, O., Lee, D., Chen, X., Goldberg, K., and Abbeel, P. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5628–5635. IEEE, 2018.
- Zhao, T. Z., Kumar, V., Levine, S., and Finn, C. Learning fine-grained bimanual manipulation with low-cost hardware, 2023.
- Zheng, Q., Henaff, M., Amos, B., and Grover, A. Semi-supervised offline reinforcement learning with action-free trajectories. In *Proceedings of the 40th International Conference on Machine Learning, ICML’23*. JMLR.org, 2023.
- Zhou, A., Kumar, V., Finn, C., and Rajeswaran, A. Policy architectures for compositional generalization in control. *arXiv preprint arXiv:2203.05960*, 2022.

- Zhu, D., Chen, J., Shen, X., Li, X., and Elhoseiny, M. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.
- Zhu, G., Xiao, X., Li, C., Ma, J., Ponraj, G., Prituja, A. V., and Ren, H. A bimanual robotic teleoperation architecture with anthropomorphic hybrid grippers for unstructured manipulation tasks. *Applied Sciences*, 10(6), 2020. ISSN 2076-3417. doi: 10.3390/app10062086.
- Zhu, X. Semi-supervised learning literature survey. *Comput Sci, University of Wisconsin-Madison*, 2005.
- Zhu, Y., Joshi, A., Stone, P., and Zhu, Y. Viola: Imitation learning for vision-based manipulation with object proposal priors. *CoRL*, 2022.
- Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.

Appendix A

Chapter 2 Appendix

A.1 Additional MTM Results

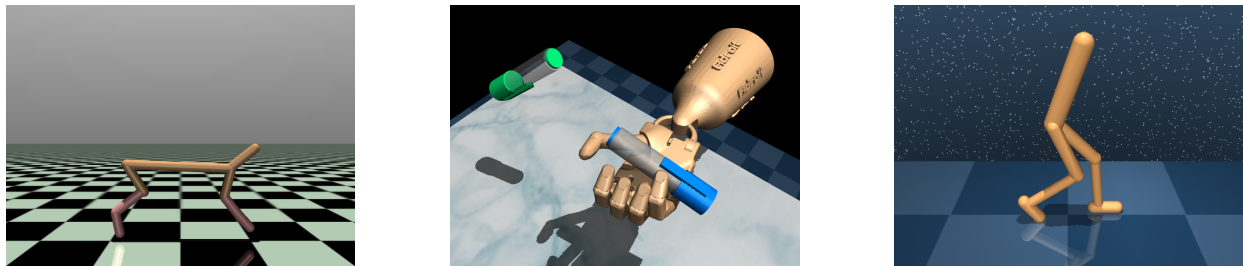
Table A.1.1: Evaluation of MTM capabilities on Adroit.

Domain	Dataset	Task	MLP	S-MTM (Ours)	MTM (Ours)
Adroit Pen	Expert	BC	62.75 ± 1.43	66.28 ± 3.28	61.25 ± 5.06
	Expert	RCBC	68.41 ± 2.27	66.29 ± 1.39	64.81 ± 1.70
	Expert	ID	0.128 ± 0.001	0.155 ± 0.001	0.331 ± 0.049
	Expert	FD	0.048 ± 0.002	0.360 ± 0.020	0.321 ± 0.048
Adroit Pen	Medium Replay	BC	33.73 ± 1.00	54.84 ± 5.08	47.10 ± 7.13
	Medium Replay	RCBC	41.26 ± 4.99	57.50 ± 3.76	58.76 ± 5.63
	Medium Replay	ID	0.308 ± 0.004	0.238 ± 0.004	0.410 ± 0.064
	Medium Replay	FD	0.657 ± 0.023	0.915 ± 0.007	0.925 ± 0.026
Adroit Door	Expert	BC	147.68 ± 0.25	149.46 ± 0.29	149.19 ± 0.72
	Expert	RCBC	148.81 ± 0.32	150.50 ± 0.14	149.93 ± 0.19
	Expert	ID	0.385 ± 0.001	0.427 ± 0.003	0.484 ± 0.024
	Expert	FD	0.199 ± 0.011	0.541 ± 0.020	0.618 ± 0.210
Adroit Door	Medium Replay	BC	27.75 ± 5.03	49.24 ± 26.85	16.30 ± 10.10
	Medium Replay	RCBC	71.51 ± 8.62	75.41 ± 8.20	51.92 ± 9.13
	Medium Replay	ID	0.532 ± 0.001	0.589 ± 0.005	0.629 ± 0.014
	Medium Replay	FD	0.976 ± 0.033	2.225 ± 0.061	2.251 ± 0.230

Table A.1.2: Evaluation of MTM capabilities on D4RL.

Domain	Dataset	Task	MLP	S-MTM (Ours)	MTM (Ours)
D4RL Hopper	Expert	BC	111.14 \pm 0.33	111.81 \pm 0.18	107.35 \pm 7.77
	Expert	RCBC	111.17 \pm 0.56	112.64 \pm 0.47	112.49 \pm 0.37
	Expert	ID	0.009 \pm 0.000	0.013 \pm 0.000	0.050 \pm 0.026
	Expert	FD	0.072 \pm 0.000	0.517 \pm 0.025	0.088 \pm 0.049
D4RL Hopper	Medium Expert	BC	58.75 \pm 3.79	60.85 \pm 3.14	54.96 \pm 2.44
	Medium Expert	RCBC	110.22 \pm 0.99	113.00 \pm 0.39	112.41 \pm 0.23
	Medium Expert	ID	0.015 \pm 0.000	0.015 \pm 0.001	0.053 \pm 0.003
	Medium Expert	FD	0.139 \pm 0.001	0.938 \pm 0.062	0.077 \pm 0.005
D4RL Hopper	Medium	BC	55.93 \pm 1.12	56.74 \pm 0.56	57.64 \pm 3.37
	Medium	RCBC	62.20 \pm 3.41	69.20 \pm 1.60	70.48 \pm 4.62
	Medium	ID	0.022 \pm 0.001	0.030 \pm 0.001	0.143 \pm 0.035
	Medium	FD	0.153 \pm 0.002	1.044 \pm 0.061	0.206 \pm 0.064
D4RL Hopper	Medium Replay	BC	35.63 \pm 6.27	36.17 \pm 4.09	29.46 \pm 6.74
	Medium Replay	RCBC	88.61 \pm 1.68	93.30 \pm 0.33	92.95 \pm 1.51
	Medium Replay	ID	0.240 \pm 0.028	0.219 \pm 0.008	0.534 \pm 0.009
	Medium Replay	FD	2.179 \pm 0.052	3.310 \pm 0.425	0.493 \pm 0.030
D4RL Walker2D	Expert	BC	109.28 \pm 0.12	108.76 \pm 0.32	107.08 \pm 1.47
	Expert	RCBC	112.21 \pm 0.31	109.83 \pm 0.58	110.08 \pm 0.82
	Expert	ID	0.021 \pm 0.000	0.055 \pm 0.001	0.233 \pm 0.038
	Expert	FD	0.077 \pm 0.001	0.233 \pm 0.012	0.177 \pm 0.031
D4RL Walker2D	Medium Expert	BC	108.45 \pm 0.31	108.49 \pm 1.00	75.64 \pm 7.78
	Medium Expert	RCBC	110.47 \pm 0.38	110.43 \pm 0.30	110.21 \pm 0.31
	Medium Expert	ID	0.019 \pm 0.000	0.038 \pm 0.001	0.213 \pm 0.030
	Medium Expert	FD	0.088 \pm 0.001	0.221 \pm 0.013	0.167 \pm 0.032
D4RL Walker2D	Medium	BC	75.91 \pm 1.87	75.87 \pm 0.44	59.82 \pm 7.06
	Medium	RCBC	78.76 \pm 2.26	78.64 \pm 2.05	78.08 \pm 2.04
	Medium	ID	0.026 \pm 0.001	0.055 \pm 0.002	0.214 \pm 0.145
	Medium	FD	0.116 \pm 0.002	0.236 \pm 0.012	0.175 \pm 0.162
D4RL Walker2D	Medium Replay	BC	23.39 \pm 2.75	48.45 \pm 2.84	21.98 \pm 2.77
	Medium Replay	RCBC	72.85 \pm 5.23	78.33 \pm 2.11	77.32 \pm 1.79
	Medium Replay	ID	0.532 \pm 0.017	0.493 \pm 0.018	0.921 \pm 0.032
	Medium Replay	FD	1.224 \pm 0.011	0.883 \pm 0.011	0.446 \pm 0.016
D4RL HalfCheetah	Expert	BC	93.14 \pm 0.16	95.21 \pm 0.44	94.19 \pm 0.21
	Expert	RCBC	94.16 \pm 0.35	95.12 \pm 0.64	94.83 \pm 0.72
	Expert	ID	0.001 \pm 0.000	0.003 \pm 0.000	0.009 \pm 0.001
	Expert	FD	0.009 \pm 0.000	0.018 \pm 0.003	0.005 \pm 0.001
D4RL HalfCheetah	Medium Expert	BC	68.04 \pm 1.57	77.88 \pm 7.21	65.73 \pm 5.69
	Medium Expert	RCBC	93.49 \pm 0.29	94.85 \pm 0.32	94.78 \pm 0.39
	Medium Expert	ID	0.001 \pm 0.000	0.001 \pm 0.000	0.012 \pm 0.002
	Medium Expert	FD	0.014 \pm 0.000	0.043 \pm 0.008	0.009 \pm 0.001
D4RL HalfCheetah	Medium	BC	42.87 \pm 0.11	43.37 \pm 0.14	43.19 \pm 0.34
	Medium	RCBC	44.43 \pm 0.26	43.83 \pm 0.22	43.65 \pm 0.08
	Medium	ID	0.001 \pm 0.000	0.005 \pm 0.000	0.027 \pm 0.017
	Medium	FD	0.020 \pm 0.000	0.053 \pm 0.011	0.020 \pm 0.010
D4RL HalfCheetah	Medium Replay	BC	36.81 \pm 0.52	39.03 \pm 0.78	19.64 \pm 11.26
	Medium Replay	RCBC	40.55 \pm 0.18	42.94 \pm 0.33	43.08 \pm 0.43
	Medium Replay	ID	0.003 \pm 0.000	0.005 \pm 0.000	0.036 \pm 0.012
	Medium Replay	FD	0.059 \pm 0.000	0.058 \pm 0.010	0.028 \pm 0.007

A.2 Additional Environment Details



(a) D4RL: HalfCheetah

(b) Adroit: Pen

(c) DM-Control: Walker2D

Figure A.2.1: **Continues Control Evaluation Settings.**

Here we provide additional details on each experiment setting. In general, our empirical evaluations are based on the standard versions of D4RL, Adroit, and ExORL. These benchmarks and setups are widely used in the community for studying various aspects of offline learning. The raw state space provided by these benchmarks typically comprise a mix of positions and velocities of different joints, bodies, and objects in the environment. We preprocess each dataset by normalizing the data before training.

D4RL (Fu et al., 2020) is a popular offline RL benchmark. As mentioned in Section 2.4.1, we test MTM on the locomotion suite of D4RL. The locomotion suite uses the *Walker*, *Hopper*, and *HalfCheetah* environments provided by OpenAI Gym (Brockman et al., 2016). We consider 4 different dataset settings: *Expert*, *Medium-Expert*, *Medium*, and *Medium-Replay*. These datasets are collected by taking trajectories of a SAC Haarnoja et al. (2018a) agent at various points in training.

Adroit (Rajeswaran et al., 2018) is a collection of dexterous manipulation tasks with a simulated five-fingered. Our MTM experiments use the *Pen*, and *Door* tasks. To match the setup of D4RL, we collect *Medium-Replay* and *Expert* trajectories for each task. This is done by training an expert policy. The *Expert* dataset comprises of rollouts of the converged policy with a small amount of action noise. The *Medium-Replay* dataset is a collection of trajectory rollouts from various checkpoints during training of the expert policy, before policy convergence. The original Adroit environment provides a dense reward and a sparse measure of task completion. For MTM experiments, we use the task completion signal as an alternative to reward, which provides a more grounded signal of task performance (a measure of the number of time steps in the episode where the task is complete).

ExORL (Yarats et al., 2022) dataset consists of trajectories collected using various unsupervised exploration algorithms. ExORL leverages dm.control developed by Tunyasuvunakool et al. (2020). We use data collected by a ProtoRL agent (Yarats et al., 2021b) in the Walker2D environment to evaluate the effectiveness of MTM representations on three different tasks: *Stand*, *Walk*, and *Run*. As the pretraining dataset has not extrinsic reward, MTM is trained with only states and actions. During downstream TD3 learning, all trajectories are relabeled with the task reward.

A.3 Model and Training Details

A.3.1 MLP Baseline Hyperparameters

Table A.3.1: MLP Hyperparameters

	Hyperparameter	Value
MLP	Nonlinearity	GELU
	Batch Size	4096
	Embedding Dim	1024
	# of Layers	2
Adam Optimizer	Learning Rate	0.0002
	Weight Decay	0.005
	Warmup Steps	5000
	Training Steps	140000
	Scheduler	cosine decay

A.3.2 MTM Model Hyperparameters

Table A.3.2: MTM Hyperparameters

	Hyperparameter	Value
General	Nonlinearity	GELU
	Batch Size	1024
	Trajectory-Segment Length	4
	Scheduler	cosine decay
	Warmup Steps	40000
	Training Steps	140000
	Dropout	0.10
	Learning Rate	0.0001
	Weight Decay	0.01
Bidirectional Transformer	# of Encoder Layers	2
	# Decoder Layers	1
	# Heads	4
	Embedding Dim	512
Mode Decoding Head	Number of Layers	2
	Embedding Dim	512

A.3.3 MTM Training Details

In this section, we specify additional details of MTM for reproduction. Numerical values of the hyperparameters are found in table A.3.1. The architecture follows the structure of (He et al., 2021) and (Liu et al., 2023a), which involves a bidirectional transformer encoder and a bidirectional transformer decoder. For each input modality there is a learned projection into the embedding space. In addition we add a 1D sinusoidal encoding to provide time index information. The encoder only processes unmasked tokens. The decoder processes the full trajectory sequence, replacing the masked out tokens with mode specific mask tokens. At the output of the decoder, we use a 2 Layer MLP with Layer Norm (Ba et al., 2016). For training the model we use the AdamW optimizer (Kingma & Ba, 2014; Loshchilov & Hutter, 2019) with a warm up period and cosine learning rate decay.

As we rely on the generative capabilities of MTM which can be conditioned on a variety of different input tokens at inference time, we train MTM with a range of mask ratios that are randomly sampled. We use a range between 0.0 and 0.6. Our random autoregressive masking scheme also requires that at least one token is predicted without future context. This is done by randomly sampling a time step and token, and masking out future tokens.

A.4 Effect of training trajectory length

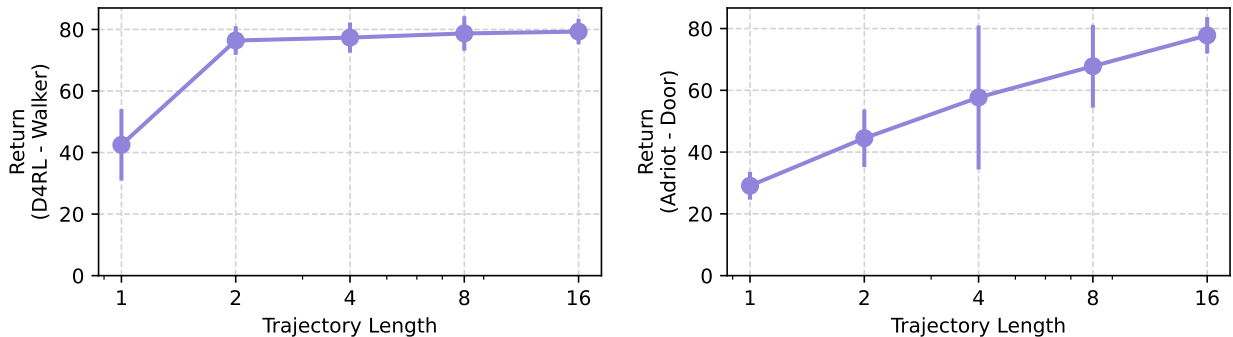


Figure A.4.1: **Effect of Trajectory Training Length.** This plot depicts the effect of changing the training trajectory length on RCBC performance, all other hyperparameters held constant. The left side shows the performance of D4RL Walker2D and the right on Adroit Door, both using their corresponding Medium-Replay Dataset.

Figure A.4.1 illustrates the effect of training trajectory length on performance. We observe that increased trajectory length has benefits in training performance. We hypothesize that with longer trajectory lengths, MTM is able to provide richer training objectives, as the model now must learn how to predict any missing component of a longer trajectory. This is especially apparent in the Adroit Door task, where we see RCBC performance increasing strongly with trajectory training length. This suggests that better results could be achieved

with longer horizon models. We see that this benefit provides diminishing returns for much longer trajectories (and additionally increases training time), which is most apparent in the D4RL Walker2D task. However, for practicality, we fix the trajectory length to 4 for all other experiments, and tune hyperparameters for this trajectory training length. Factors such as mask ratio could be tuned to optimize performance and training time for longer trajectory lengths, but we leave this exploration for future work.

A.5 Additional plots

A.5.1 Masking Patterns

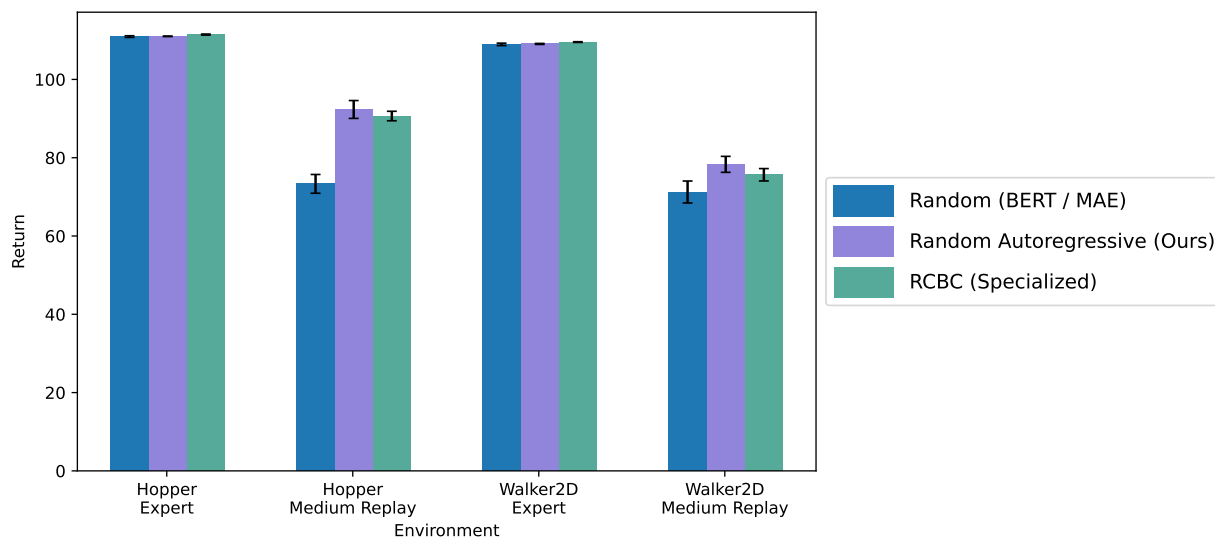


Figure A.5.1: **Impact of Masking Patterns.** This plot shows MTM RCBC performance trained with three different masking patterns, random, random autoregressive, and a specialized RCBC mask. This is a repeat of Figure 2.4, except the Y-axis is unscaled.

A.5.2 Heteromodal MTM

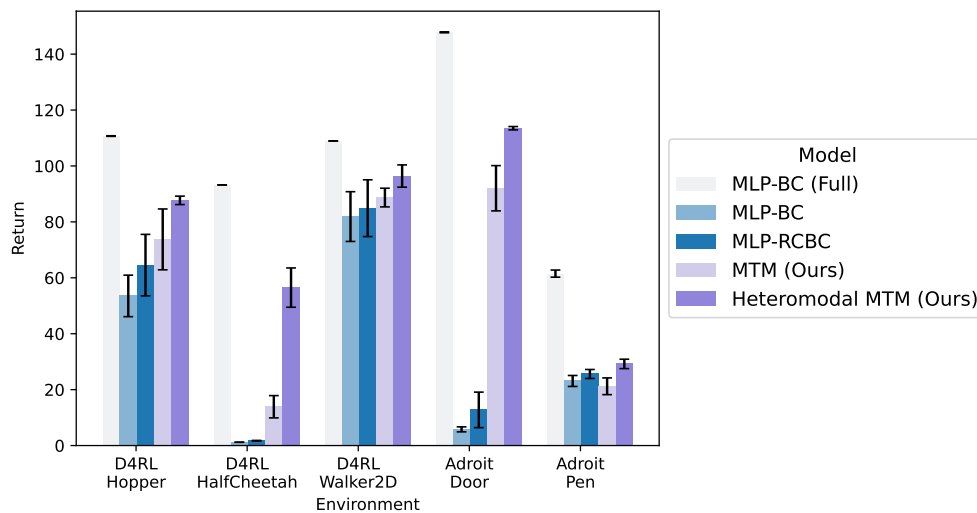


Figure A.5.2: **MTM can effectively learn from heteromodal datasets.** This figure, which shows the performance of our Heteromodal MTM model, is a repeat of Figure 2.5, except the Y-axis is unscaled. Instead we observe the absolute return for each environment. In addition we provide the performance of BC trained on the entire training set (95% of the provided dataset) as reference for the **oracle** performance that can be achieved.

A.5.3 Representation Learning

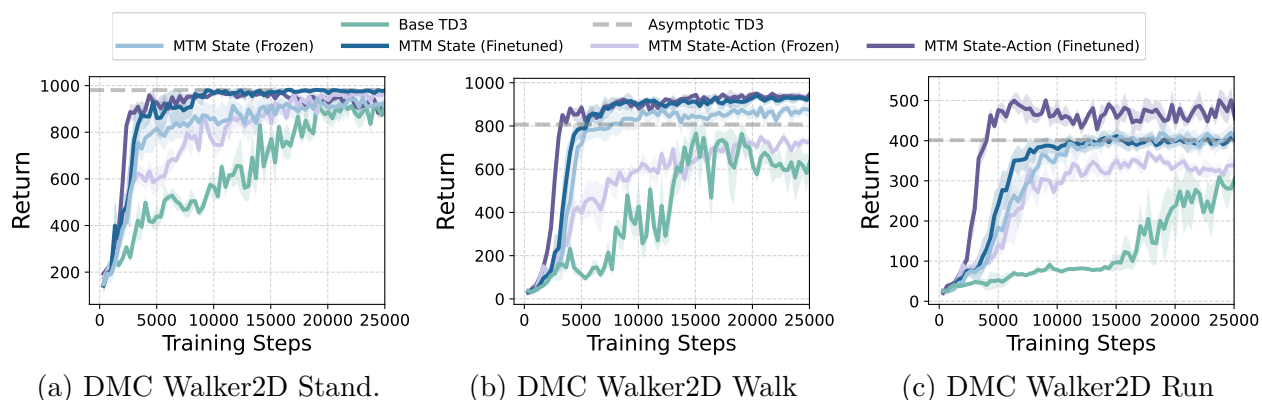


Figure A.5.3: **Finetuned and frozen MTM representations.** Here we additionally provide the learning curves for frozen MTM representations on top of those provided in Figure 2.7. Both frozen MTM features and finetuned MTM features enable faster learning, but we do see that finetuning offers the best learning benefits across tasks.

Appendix B

Chapter 3 Appendix

B.1 Appendix

B.1.1 Hyperparameters

In both experiments the image observations are first encoded with a 5 layer CNN with ReLU activations. The CNN encoder is shared across embedding the current state and the demonstration trajectory. The policy network is also the same, which consists of 3 FiLM blocks using GELU nonlinearities and using 128 hidden units per layer. For all experiments that use the contrastive objective across paired trajectories, a weighting of 10 is used on the contrastive loss. For all experiments using pseudo-labeled trajectories, a weighting of 0.5 is used on the imitation loss with pseudo-labeled trajectories.

Semantic Goal Navigation For this task we adopt the oracle trajectory encoder, which takes the final frame of the trajectory (which fully specifies the task) and encodes it with CNN. The images are 64x64. The policy is trained with a learning rate of 1e-3 with 4000 warm up steps. Frame stacking of 2 is employed on the observations. For each experiment we train for 200k iterations.

For the self supervised augmentations, we employ random resizing, cropping, horizontal flip, and vertical flip. An additional one layer projection is applied before applying the self supervised contrastive loss, which we employ with a weight of 0.05.

Sequential Goal Navigation For this task, we make no assumptions on what frames are important and use a bidirectional transformer that attends over all states in the trajectory. The transformer has 2 hidden layers and 2 attention heads, and the goal encoding is extracted with an additional class token. Images are 16x16. We use a learning rate of 3e-4 and train for 60k iterations.

B.1.2 Effect of more pseudo label pair value k

Here we study how choosing different values of k and different iterations of relabeling effect final performance. In the pseudo-labeling stage, we fix k controls the number of possible pairs each unlabeled trajectory can use for training. In addition, we experiment with using the student model as a teacher model for one additional iteration of training. We use the Semantic Goal reaching task, with 15% of the full dataset size. The results are summarized in Table B.1.1. Results are mixed overall, and it seems the exact choice of k does not have a significant impact on results. In addition it seems that repeating the pseudo labeling process for additional iterations does not have any significant gains on performance. This could be due to the simplicity of the task, as well as the already high trajectory retrieval scores across all k . We suspect that for more difficult tasks, these parameters will have a more significant impact on final performance.

Table B.1.1: Iterative Relabeling on Semantic Goal Navigation

Iteration	Test Success rate % with $k=$			
	10	50	100	200
1	82 ± 1.4	88.7 ± 6.9	83.3 ± 17.4	95.4 ± 1.2
2	88 ± 2.1	87.5 ± 4.5	78.5 ± 2.5	86.5 ± 0.5

Appendix C

Chapter 4 Appendix

C.1 Adaptation

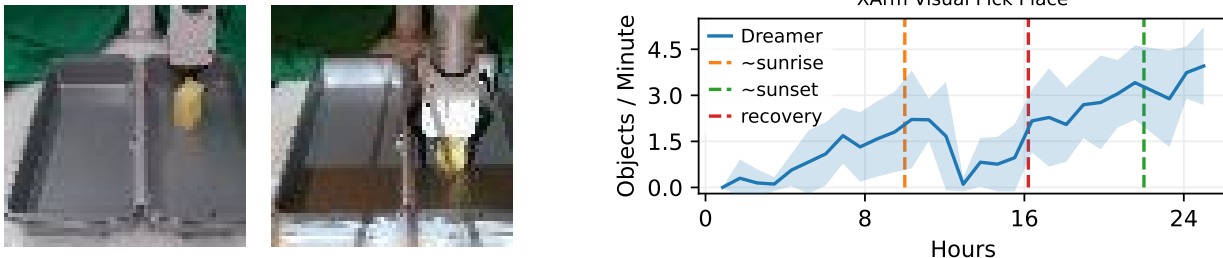


Figure C.1.1: The left two images are raw observations consumed by Dreamer. The leftmost image is an image observation as seen by the XArm at night, when it was trained. The next image shows an observation during sunrise. Despite the vast difference in pixel space, the XArm is able to recover, and then surpass, the original performance in approximately 5 hours. Even after 24 hours when the lighting shifts to night time conditions, the XArm is able to maintain performance.

Real world robot learning faces practical challenges such as changing environmental conditions and time varying dynamics. We found that Dreamer is able to adapt to the current environmental conditions with no change to the learning algorithm. This shows promise for using Dreamer in continual learning settings (Parisi et al., 2019). Adaptation of the quadruped to external perturbations is reported in Section 4.3.1 and Figure 4.5.

The XArm, situated near large windows, is able to adapt and maintain performance under the presence of changing lighting conditions. The XArm experiments were conducted after sundown to keep the lighting conditions constant throughout training. Figure C.1.1 shows the learning curve of the XArm. As expected, the performance of the XArm drops during sunrise. However, the XArm is able to adapt to the change in lighting conditions in about 5

hours time and recover the original performance, which is faster than it would be to train from scratch. A careful inspection of the image observations at these times, as shown in Figure C.1.1, reveals that the robot received observations with strong light rays covering the scene which greatly differs from the original training observations.

C.2 Imagination

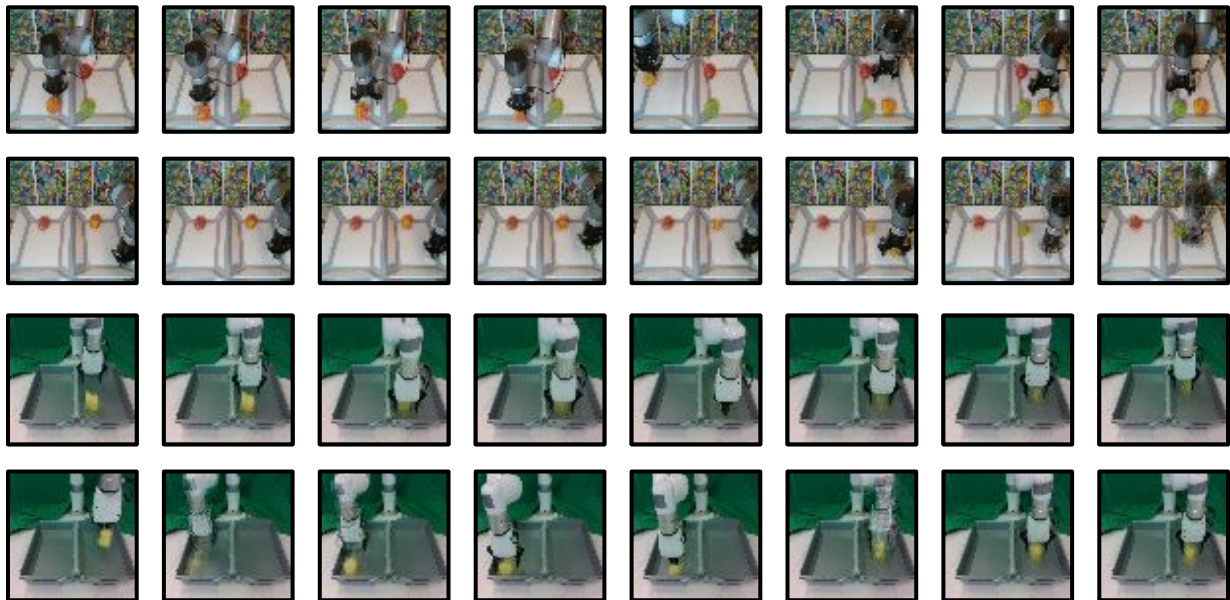


Figure C.2.1: To introspect the policy, we can roll out trajectories in the latent space of Dreamer, then decode the images to visualize the intent of the actor network. Each row is an imagined trajectory, showing every 2nd frame. **Top:** Latent rollouts on the UR5 environment. Multiple objects introduce more visual complexity that the network has to model. Note the second trajectory, which shows a static orange ball becoming a green ball. **Bottom:** Latent rollouts on the XArm environment.

C.3 Detailed Related Work

RL for locomotion A common approach is to train RL agents from large amounts of simulated data under domain and dynamics randomization (Peng et al., 2018; Lee et al., 2020; Rudin et al., 2021; Siekmann et al., 2021; Escontrela et al., 2022; Miki et al., 2022; Kumar et al., 2021; Rusu et al., 2016; Bohez et al., 2022), then freezing the learned policy and deploying it to the real world. Smith et al. (2021) explored pre-training policies in simulation and fine-tuning them with real world data. Yang et al. (2019) investigate learning a dynamics model using a multi-step loss and using model predictive control to accomplish a specified task. Yang et al. (2022a) train locomotion policies in the real world but require a recovery controller trained in simulation to avoid unsafe states. In contrast, we use no simulators or reset policies and directly train on the physical robot. While prior work in locomotion has successfully learned walking behaviors in the real world, these works generally required several domain-specific assumptions or pretraining with simulators. Ha et al. (2020a) achieved successful walking on the Minitaur robot in 90 minutes. However, the authors manually programmed a reset policy that was used when the robot fell on its back, while in our work the robot must learn to flip over and stand up. Additionally, the Minitaur robot is simpler than the A1 as it has 8 actuators compared to 12 on the A1. In recent work, Smith et al. (2022) utilize a high update-to-data ratio (UTD) RL algorithm to learn walking from 20 minutes of robot training data. However, their work assumes the availability of a reset policy and therefore comprises of a different learning problem compared to the problem we tackle of learning to flip over and walk from scratch. Additionally, we show our approach generalizes to environments with image observations and sparse rewards.

RL for manipulation Learning promises to enable robot manipulators to solve contact rich tasks in open real world environments. One class of methods attempts to scale up experience collection through a fleet of robots (Kalashnikov et al., 2018, 2021; Ebert et al., 2021; Dasari et al., 2019; Levine et al., 2018). In contrast, we only leverage one robot, but parallelize an agent’s experience by using the learned world model. Another common approach is to leverage expert demonstrations or other task priors (Pinto & Gupta, 2015; Ha & Song, 2021; Xie et al., 2019; Schoettler et al., 2019; Sivakumar et al., 2022). James & Davison (2021); James et al. (2021) leverages a few demonstrations to increase the sample-efficiency of Q learning by focusing the learner on important aspects of the scene. Other approaches, as in locomotion, first utilize a simulator, then transfer to the real world (Tzeng et al., 2015; Akkaya et al., 2019; OpenAI et al., 2018; Irpan et al., 2020). Our work focuses on single-robot environments where the agent must learn through a small amount of interaction with the world. Meanwhile, the Google Arm Farm line of work by Levine et al. leverages over 580k grasp attempts gathered by 7 robots and collected over 4 months. We believe that a method such as Dreamer could benefit greatly from this scale of training data, however it is unlikely that works such as MT-OPT/QT-OPT Kalashnikov et al. (2018, 2021) would work well in the low data regime that Dreamer excels in.

Model-based RL Due to its higher sample-efficiency over model-free methods, model-based RL is a promising approach to learning on real world robots (Deisenroth et al., 2013). A model based method first learns a dynamics model, which can then be used to plan actions (Nagabandi et al., 2019; Hafner et al., 2019b; Chua et al., 2018; Nagabandi et al., 2017; Becker-Ehmck et al., 2020), or be used as a simulator to learn a policy network as in Dreamer (Hafner et al., 2019a, 2020). One approach to tackle the high visual complexity of the world is to learn an action conditioned video prediction model (Finn & Levine, 2017; Ebert et al., 2018; Finn et al., 2016a). One downside of this approach is the need to directly predict high dimensional observations, which can be computationally inefficient and easily drift. Dreamer learns a dynamics model in a latent space, allowing more efficient rollouts and avoids relying on high quality visual reconstructions for the policy. Another line of work proposes to learn latent dynamics models without having to reconstruct inputs (Deng et al., 2021; Okada & Taniguchi, 2021; Bharadhwaj et al., 2022; Paster et al., 2021; Li et al., 2021), which we see as a promising approach for supporting moving view points in cluttered environments.

C.4 Environment and Hardware Details

For every robot setup that involved vision (UR5, XArm, Sphero), we used a RealSense D435 camera positioned to offer a fixed 3rd person view of the scene.

A1 We used the A1 quadrupedal robot by Unitree. The RL policy outputs actions at a frequency that is too high for the PD controller to track, which we overcome by lowpass filtering the action sequence. The joint range allows the legs to self-collide with the body, which can be damaging to the motors and increase battery consumption. We limited the joint range to decrease self-collisions. Finally, the EKF velocity estimator relies on foot-ground contact events to prevent significant drift in the estimates, so we employ a curriculum reward function that does not reward the robot for forward velocity until the robot is upright with extended legs. We also designed a shell which we 3D printed in order to better protect the cables and hardware and provide a smoother rolling over.

XArm & UR5 We utilized slanted bins to prevent objects from leaving the work area during the long-running pick and place experiments on the UR5, which is common practice Levine et al. (2018); Kalashnikov et al. (2018). We also added a partition behind the setup to keep the background constant. It would be interesting to study how a gripper-mounted camera would impact policy performance Hsu et al. (2022), however we report strong results without this design choice. For the XArm we use the uFactory xArm Gripper. For the UR5, we use the Robotiq 2F-85 parallel jaw gripper. The bin locations are predetermined and provided as part of the environment to prevent the robot from colliding with the bin. In addition, movement in the Z axis is only enabled while holding an object and the gripper automatically opens once above the other bin.

Sphero We used a rectangular enclosure of $0.8 \times 0.8\text{m}^2$ to keep the sphero robot within the camera view. We used a simple OpenCV script to estimate the L2 distance between the Sphero and the goal position to provide a dense reward for policy optimization. This positional information was not provided to the agent, which it had to learn from the raw top-down images.

C.5 Hyperparameters

Name	Symbol	Value
General		
Replay capacity (FIFO)	—	10^6
Start learning	—	10^4
Batch size	B	32
Batch length	T	32
MLP size	—	4×512
Activation	—	LayerNorm + ELU
World Model		
RSSM size	—	512
Number of latents	—	32
Classes per latent	—	32
KL balancing	—	0.8
Actor Critic		
Imagination horizon	H	15
Discount	γ	0.95
Return lambda	λ	0.95
Target update interval	—	100
All Optimizers		
Gradient clipping	—	100
Learning rate	—	10^{-4}
Adam epsilon	ϵ	10^{-6}

Appendix D

Chapter 6 Appendix

D.1 Additional Real-world Task Visualization

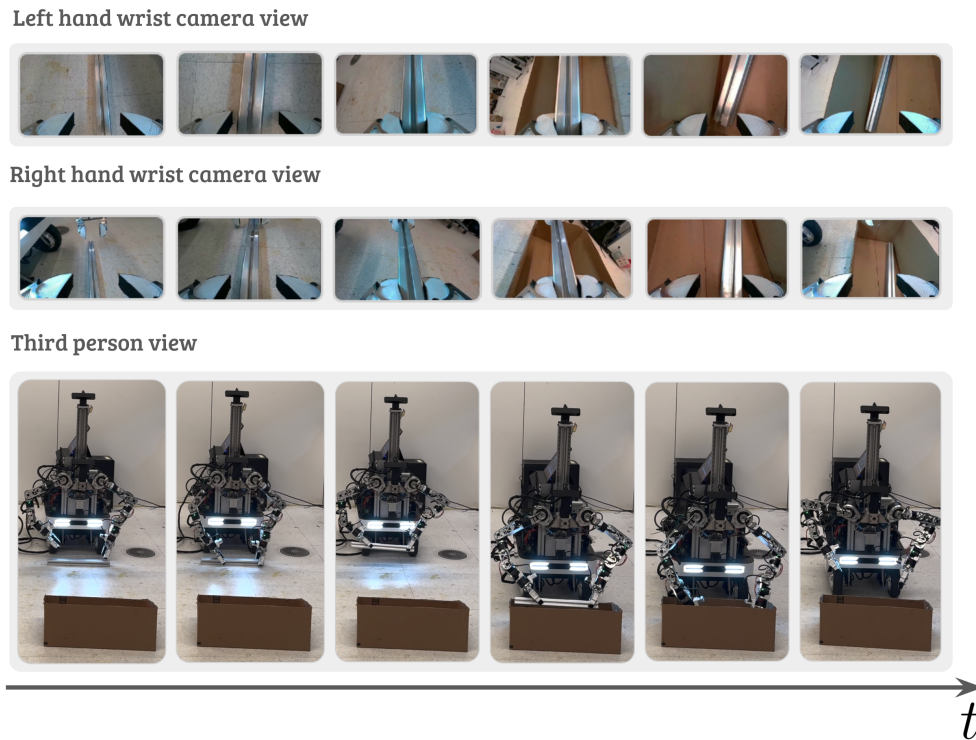


Figure D.1.1: A step-by-step illustration of the industrial part transport task. The robot first needs to pick up the heavy industrial part from the floor using both hands. It then searches for and moves towards the tote. Finally, the robot must accurately drop the industrial part into the tote. The first two rows display the left and right wrist camera views along one autonomous trajectory. The bottom row shows a third-person view of the entire robot system during task execution.

D.2 Simulation Task Visualizations

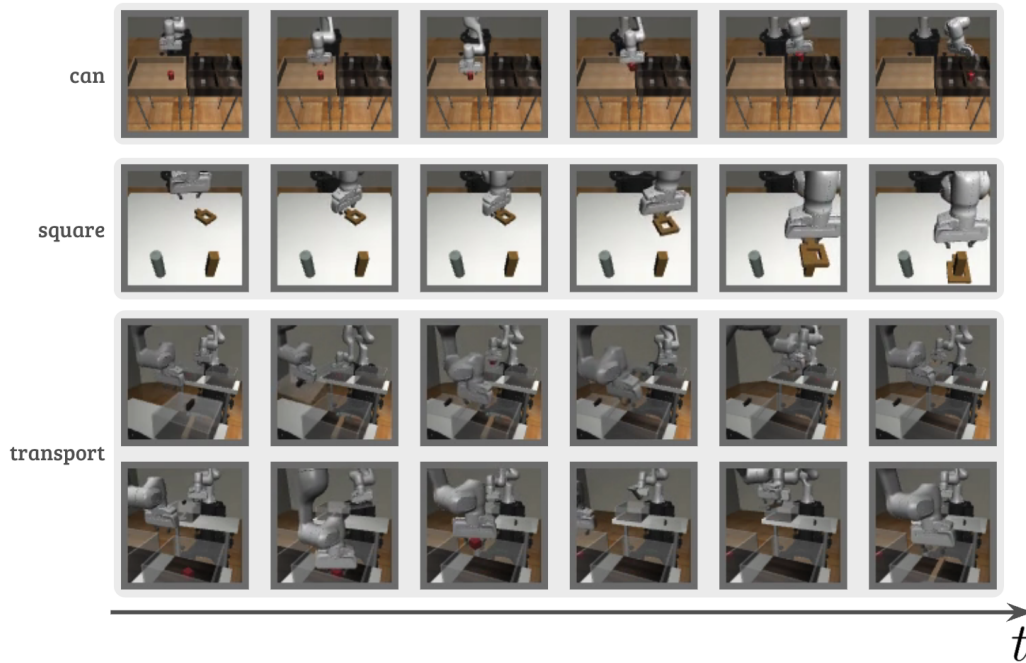


Figure D.2.1: An illustration of the three manipulation tasks we use in Robomimic (Mandlekar et al., 2021): Pick up a can and place it in a bin; insert a square object into a pole; coordinate two arms to transport a tool.

D.3 Robot Hardware Details

The design space for such a complex system is large. Thus we utilize the following three key principles the development for our RoboCopilot system:

- **Safe:** The robot hardware must be safe for humans, the environment, and itself. Compliance, a mechanical property that describes the ability of the robot to respond to external forces, is critical to achieving this safety.
- **Easy to Use:** To facilitate efficient interactive learning, our system needs a capable teleoperation system that allows human operators to perform complex and contact-rich tasks. Additionally, this system must enable the human operator to take control of the robot seamlessly when necessary.

- **Accessible:** Our design aims to be cost-efficient and simple while still being able to perform a wide range of daily tasks. We believe accessibility is crucial for scaling up data collection and robot deployment in real-world scenarios.

Our robot, guided by the key principles above, is a custom low-cost mobile bimanual manipulator designed for everyday tasks. It consists of 7 DOF arms with parallel jaw grippers, an actuatable torso, and an omnidirectional base, totaling 20 degrees of freedom. We use two realsense D405 cameras Intel (2024) mounted on the wrist which are used to provide camera observations to the robot.

Compliance from Quasi-Direct Drive Actuators. Compliance allows the robot to feel when it is perturbed by an external force, which is crucial for the robot to safely work with humans and perform complicated tasks in environments where there is high uncertainty. Compliance can be achieved with the addition of force torque sensors by measuring and responding to external forces in a tight control loop Pratt & Williamson (1995); Albu-Schäffer et al. (2007). Compliance can also be achieved passively through a backdrivable transmission, where the system can be easily driven by external force without active control Wyrobek et al. (2008); Gealy et al. (2019); Quigley et al. (2011). Taking cost and simplicity into consideration we achieve compliance with off-the-shelf mini-cheetah style Katz (2018a) quasi direct drive (QDD) transmissions which achieve backdrivability through a low gear ratio planetary gearbox. In this setup, external forces applied at the gearbox output are transmitted to the motor and can be detected by measuring motor currents, where the motor can effectively act as a torque sensor.

Actuator Selection. We use off-the-shelf mini-cheetah style Katz (2018a) brushless actuators with integrated planetary gearboxes Quasi Direct Drive (QDD) actuator as our robot joint modules. These QDDs typically have gear reduction ratios smaller than 10. With high torque brushless motors, these actuators provide sufficient torque for robotic applications while maintaining backdrivability. Although planetary gearboxes have backlash compared to gearboxes such as harmonic drives and cycloid drives, which may reduce precision, recent work utilizing end-to-end imitation learning with tight feedback loops suggests that low precision hardware can still be used for fine-grained tasks with feedback control Myers et al. (2023); Fang et al. (2023c). Furthermore, with the rise of legged robots applications, there are now widely available, affordable, and high-performing QDD options on the market. We are using 20Nm gear ratio of 9 actuators for the shoulder joints and 12Nm gear ratio of 40 actuators for the upper arm rotation and elbow joint and 3Nm gear ratio 10 actuators for the lower arm joints as well as the gripper joints, giving an approximate continuous payload of 1kg. The total cost of all 8 QDDs on each arm cost less than \$2000 for our design. For a more detailed cost breakdown, see Table D.3.1.

Arm Design. Figure D.3.1 Shows the dimensions of the robot and the teleoperation device. The specs of our robot and teleoperation device are shown in Table D.3.2. The bimanual system consists of replicating the robot arm and teleoperation device. We mount them at a 45 degree angle so that the first 2 joints can share the load of gravity. Following (Wu et al., 2023e), we build a scaled kinematic replica of the arm as a teleoperation device

Table D.3.1: Robot cost breakdown per arm. Our 7 degrees of freedom arm consists of two 20Nm gear ratio 9:1 QDD(quasi-direct-drive) motors as the two shoulder actuators, two 12Nm gear ratio 40:1 QDD as the upper arm rotate and the elbow actuators and three 3Nm gear ratio 10:1 QDD for the lower arm rotate and the wrist yaw and pitch joints.

Part Name	QDD 20nm 9:1	QDD 12nm 40:1	QDD 3nm 10:1	Connectors	Total Price
Price Each	\$320	\$126	\$95	\$634	
Quantity	2	2	3	1 set	
Price	\$640	\$252	\$285	\$634	\$1811

and use the same motors we used for the robot arm for the teleoperation device. We set the scale to be 70% of the original size for easier human teleoperation. This however results in a design constraint, as the distance between joint 6 and joint 7 is 60mm, and with a motor diameter of 57mm, there leaves almost no room to scale down the teaching device. For this DOF we did not scale the kinematics, and found that despite not matching exactly, still resulted in a very capable and intuitive device.

Table D.3.2: Various specifications about the robot and teleoperation device.

Robot	
Mass	4.6kg
Reach (Full Range)	572mm
Upper Arm Length	279mm
Lower Arm Length	252mm
Payload continuous	~ 1.5kg
Payload peak	~ 3.5kg
Degrees of Freedom	7
Teleoperation Device	
Mass	2.6kg
Upper Arm Length	195mm
Lower Arm Length	191mm
Degrees of Freedom	7

Gripper Design. Our high-speed gripper features an offset slider-crank linkage design. This design was chosen for simplicity and robustness over other low cost hands which have more complex mechanisms such as those presented by Guo et al. (2019b); Ma & Dollar (2017); Shaw et al. (2023). The finger is designed to grasp on everyday sized objects. We implement layer jamming, leveraging 3D the printer slicer to generate Finray like geometry from our simple finger shape Elgeneidy et al. (2019); Crooks et al. (2016); Shan & Birglen (2023). This

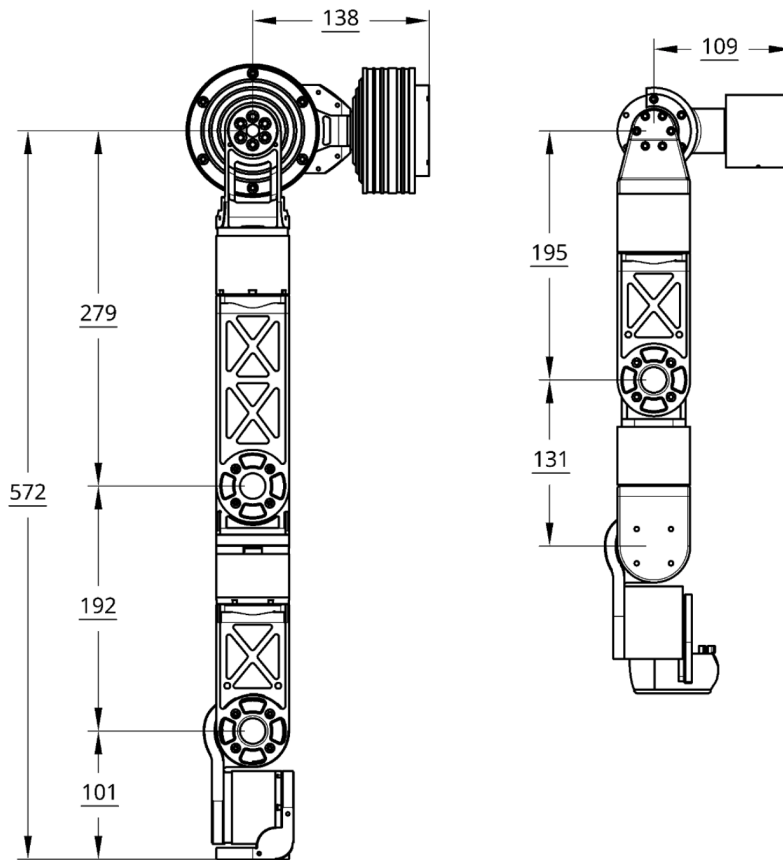


Figure D.3.1: A to scale drawing of our robot arm (left) and teleoperation device (right), with dimensions shown in millimeters.

design allows for efficient and reliable grasping of various objects. We leverage the same QDD actuators as our arm, which have a nominal RPM of 120, enabling us to achieve a continuous max gripping force of 25 N while being able to close and open under 1 second.

Mobility. A fully self contained mobile robot solution is important for ease of use and deployment in real world situations as well as the massive expansion of the robots workspace. Oftentimes, mobility in manipulation is achieved through using a mobile robot base, and mounting the rest of the robot system on top of it Holmberg & Khatib (2000); Wu et al. (2023b); Wyrobek et al. (2008). In line with the omni-directional base setup described in Xiong et al. (2024), we integrated our system with the off the shelf AgileX Ranger Mini 2 base, facilitating mobile bimanual manipulation. To expand the robot’s vertical operational space and enable the robot to pick up objects on the ground, the bimanual robot is mounted on a gantry, enabling vertical movement. The robot includes an onboard computer and router for Ethernet connectivity without Internet access. The entire mobile platform is powered by the Ranger Mini 2’s internal battery and an onboard 24v battery to make it truly wireless.

D.4 Teleoperation Capabilities

Our proposed RoboCopilot system not only enables efficient interactive teaching but also enhances teleoperation capabilities beyond the unilateral teleoperation devices Zhao et al. (2023); Wu et al. (2023e). This improvement is achieved through active joint motor selection, which provides inertia compensation and bilateral feedback. Additionally, our fully compliant arm design allows for contact-rich bimanual tasks without compromising safety or risking damage to the robot hardware. In Figure D.4.1 below, we demonstrate several daily kitchen tasks performed via teleoperation to showcase our system’s capabilities.

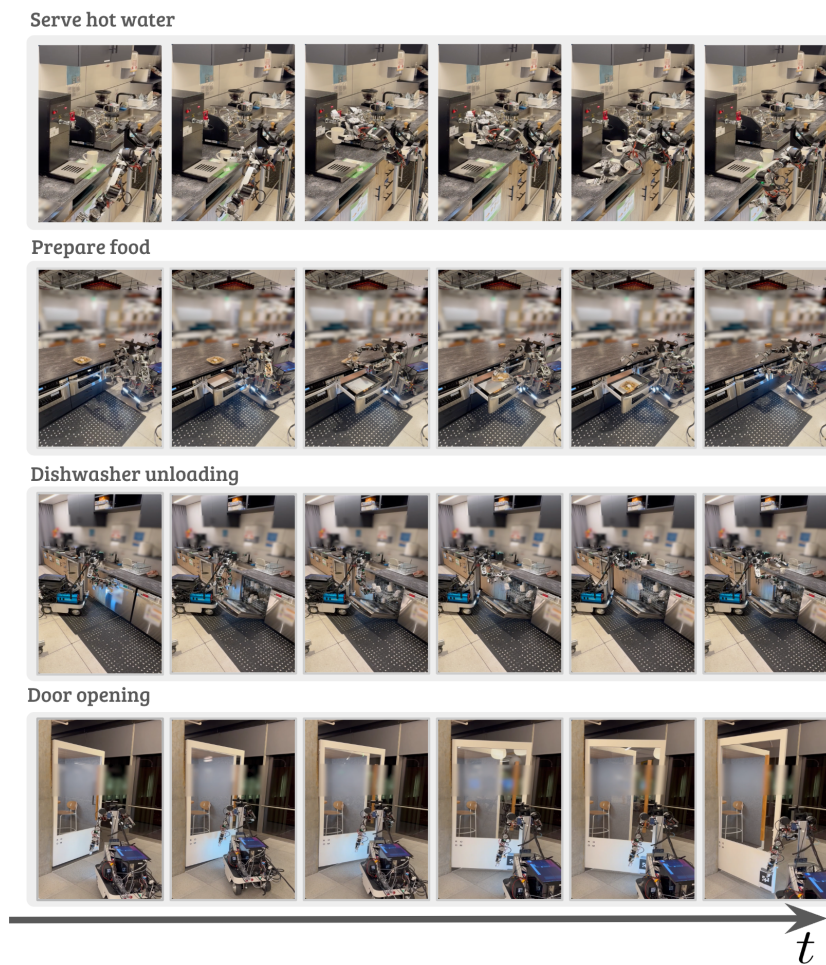


Figure D.4.1: Teleoperated Execution of Various Contact-Rich and Whole-Body Control Tasks. The RoboCopilot system demonstrates its proficiency in performing a range of daily kitchen tasks under human teleoperation. These tasks include serving hot water, preparing food, unloading the dishwasher, and opening doors. The robot’s fully compliant design enables safe and effective handling of contact-rich interactions, showcasing its ability to perform a wide range of complicated tasks.

Appendix E

Chapter 8 Appendix

E.1 GPT-4V Prompting Details

GPT-4/GPT-4V is often used zero-shot in robotic applications, due to its strong general understanding. We tuned our prompt for the language table tasks to achieve the best possible performance, using various prompting strategies. The prompt is seen in detail below in Figure E.1.1. We use a descriptive task prompt, Set-of-Mark (Yang et al., 2023), structured outputs, in context examples, and chain of thought prompting (Wei et al., 2023).

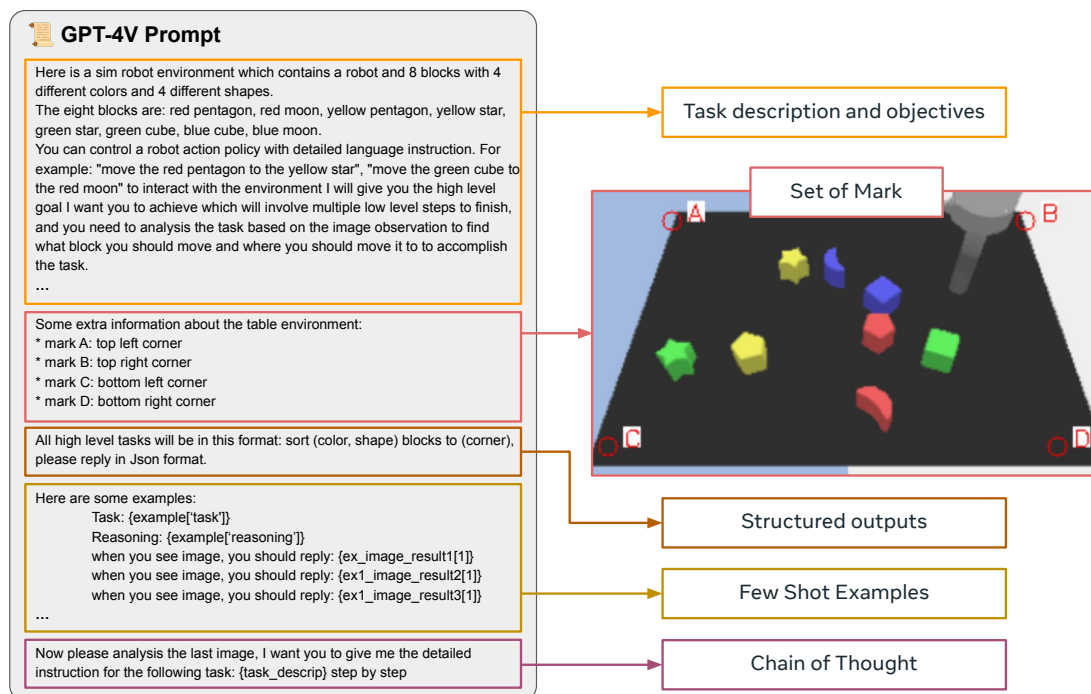


Figure E.1.1: Prompt for using GPT-4V on Language Table as a pretrained VLM. “...” indicates truncation for brevity, but follows the rest of the text in the section.

E.2 Dataset Details For Language Table

Language table contains a low level text description for each trajectory. We convert this data into chat-like interactions using programmatic templates, common in language based robotics (Lynch & Sermanet, 2021; Shridhar et al., 2021; Jiang et al., 2023). Below in Figure E.2.1, we provide the pseudocode with (truncated) examples for how we generate chat question answer pairs for training.

```

QUESTION_LIST = ["Can you control the robot to {instruction}?",
                 "Can you {instruction}?",
                 "Please {instruction}.",
                 "Given the current observation, how can you {instruction}?."]

def followup():
    start = np.random.choice([None, "first, ", "please, "])
    verb = np.random.choice(["explain", "verbalize"])
    core = np.random.choice(["how you would accomplish this task",
                             "the desired action",
                             "the next step you are going to do"])
    act = np.random.choice(["before acting", "prior to acting"])
    if start is None:
        sentence = verb + " " + core
    else:
        sentence = start + verb + " " + core
    if np.random.rand() > 0.5:
        sentence = sentence + " " + act + "."
    else:
        sentence = act + ", " + sentence + "."
    sentence = sentence[0].upper() + sentence[1:]
    return sentence

def process_instruction(instruction_string, use_extra=True):
    i_string = instruction_string.lower()
    question = np.random.choice(QUESTION_LIST).format(instruction=i_string)
    if use_extra:
        extra_instruction = followup()
        question = question + " " + extra_instruction
    return question

ANSWER_LIST = ["Sure, [ACT].", "[ACT].", "Let's move the robot [ACT]."]
ANSWER_DETAILED_LIST = ["I will {detailed_instuction} [ACT].",
                        "Sure, I will {detailed_instuction} [ACT].",
                        "I should {detailed_instuction} [ACT]."]

def process_ans_and_ques(instruction_string):
    # sometimes add more details to instruction and prvide mode details to the answer
    if np.random.rand() > 0.8:
        question = process_short_horizon_instruction(instruction_string, use_extra=True)
        answer = np.random.choice(ANSWER_LIST)
        answer = answer.format(detailed_instuction=instruction_string)
    else:
        question = process_short_horizon_instruction(instruction_string, use_extra=False)
        answer = np.random.choice(PLANNER_ANSWER_LIST)
    return question, answer

```

Figure E.2.1: Example programmatic generation for LCB training with language table data.