The Dissertation Committee for Albert Yu
certifies that this is the approved version of the following dissertation:

Augmenting Robotic Manipulation through Natural Language

Committee:

Raymond Mooney, Co-supervisor

Roberto Martín-Martín, Co-supervisor

Peter Stone

Yonatan Bisk

Augmenting Robotic Manipulation through Natural Language

by

Albert Yu

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2026

Acknowledgments

I had the privilege of being a student of two tremendous advisors. From Ray, I learned to think big, be creative, and to attempt to live out the values of academia—contribute to knowledge, be principled, keep an open mind, challenge your own assumptions, and try to leave the world a bit better. From Roberto, I grew significantly as a researcher, learning and improving in almost every aspect of the research process, from proposing creative new project ideas, to testing out things quickly, to conveying and presenting information in papers, talks, posters, and figures in a very logical and digestible manner. Being a product of both advisors has been a very fulfilling experience that I will treasure for the rest of my life.

In addition to my advisors, I also thank my committee members, Profs. Peter Stone and Yonatan Bisk, for their very valuable feedback and thought-provoking questions during the proposal and defense. I have had great respect for their excellent, broad body of research since before my PhD, and it was a great honor to have them on my committee.

I want to thank my collaborators for each chapter. In addition to my advisors, I deeply thank Adeline Foote for her work on results that supported some of Chapter 3 and a lot of Chapter 4. During the course of Chapter 5, Chengshu Li provided amazing methodological ideas, Luca Macesanu and Arnav Balaji helped a lot in user studies and paper writing, and Ruchira Ray developed a great web demo. I also thank the roughly 20 user study participants for their generosity of time. In Chapter 6, I owe a big debt of gratitude to Will Reger, Cosmo Wu, and Da Liu for help on implementation, paper figures and writing, and a large portion of the experimental trials. Most of these students were undergrads when I worked with them. They were extraordinarily dedicated, and I look forward to reading their dissertations in the future.

I also want to thank the many research collaborators on projects we worked on

that did not make it into this thesis. I had a chance to learn a lot from these projects that helped expand my view and inspired me to think of new research directions. These folks include Rutav Shah, Yifeng Zhu, Mina Huh, Huihan Liu, Jonathan Yang, Homer Walke, Avi Singh, Yanjay Orbik, Zhanpeng He, Sergey Levine, Jiajun Wu, Yuke Zhu, Amy Pavel, Maya Cakmak, Jane Shi, Fan Wang, Michael Schultz, Brad Knox, Shasank Govindarajan, Natalia Cislo, Mary Thompson, Michael Baldea, Carlos Landaverde-Alvarado, Bruce Eldridge, Ayub Lakhani, Mark Nixon, Claire Wright, Beth Hood, Erik Frey, Baruch Tabanpour, Daniel Freeman, Shuozhe Li, Steven Shi, and Soroush Nasiriany.

I also want to thank my labmates for many fruitful discussions, conversations, feedback, and our friendship. In Ray's lab, it was great to be around Jierui Li, Jordan Voas, Vanya Cohen, Yili Wang, Prasoon Goyal, Jialin Wu, and Sheena Panthaplackel. In Roberto's lab, I had a wonderful time with Rutav Shah, Jiaheng Hu, Arpit Bahety, Shivin Dass, Bowen Jiang, Sateesh Kumar, Chris Huang, Ben Abbatematteo, Junhong Xu, Skand Peri, Srinath Tankasala, Gu-cheol Jeong, Priyanka Mandikal, Taijing Chen, and Danny Tran. Both labs were very supportive, collaborative, and thought-provoking, and it was a great environment to do five years of research in.

I also want to thank all of the mentors, teachers, collaborators, and professors that helped raise me up to be a doctor. I might not have done research had Prof. Sergey Levine not accepted me into his lab after what I thought was a grilling of an interview back when I barely knew what a CNN was. I owe much of my foundational AI, deep learning, and RL knowledge to Sergey through three courses I took with him, as well as three undergraduate research projects. I also want to thank the postdocs, PhDs, and research staff in his lab who I had a great pleasure of collaborating with, including Homer Walke, Avi Singh, Aviral Kumar, Nick Rhinehart, and Yanjay Orbik, as well as the undergrads I worked closely with, including Jonathan Yang, Huihan Liu, Larry Yang, Gaoyue Zhou, and Jesse Zhang.

I also want to thank all of my college professors, internship managers, mentors,

and collaborators, for much of what I have learned as an adult that has contributed a lot to my engineering and research competence. I also believe that a handful of incredible teachers during my K-12 education helped foster my interest in math and science, and my curiosity in general that helped shape the course of my life as a lifelong learner.

I also want to thank the professors and PhDs whom I had the pleasure of TAing for, as I felt that undergrad TAing and responding to many students' questions and complaints was a great training ground for addressing anonymous reviewers' questions and concerns during rebuttals. These include Profs. Stuart Russell, Dawn Song, Satish Rao, Anca Dragan, and Lecturer Nathan Lambert.

Working hard in the background were department staff that helped keep everything running smoothly. I want to thank Catherine Vu, Anai Moreno, Zarko Vukovich, Megan Booth, Tiffany Guridy, Amy Bush, April Cafaro, Patrick Goetz, Gabrielle Bouzigard, Tyra Person, Joe Trent, and Shane Williams for helping me in one form or another in carrying out my research duties.

I want to thank my friends for keeping me sane during the course of my PhD. It is a fool's errand to try to provide an exhaustive list, but in addition to the folks listed above, I want to express my appreciation to: Zhisheng Zheng, Michael Zheng, Lucy Zhao, Jesse Zhang, Arthur Zhang, Fangcong Yin, Michael Yeh, Mesut Yang, Daniel Wu, Howard Wang, Manya Wadhwa, Sid Verma, Aditya Vaidya, Roger Tseng, Ray Tseng, Ryan Tran, Brandon Trabucco, Anirudh Srinivasan, Zayne Sprague, Ian Shih, Steven Shi, Chang Shi, Siqi Shang, Yotam Sechayk, Saagar Sanghavi, Hyeonggon Ryu, Max Rudolph, Juan Diego Rodriguez, Tristan Ramon, Carl Qi, Puyuan Peng, Kevin Nguyen, Henry Nachman, Jason Liu, Shuijing Liu, Huihan Liu, Michael Liu, Leo Liu, Kevin Lin, Allan Li, Cathy Li, Sasha Khazatsky, Gauri Khambhatla, Vishnu Iyer, Akhil Gharpurey, Anuj Diwan, Ashwin Devaraj, Kwanghee Choi, Jeffrey Chen, Hung-Ting Chen, Eric Chang, Desmond Celo, Maulik Bhatt, Karim Benharrak, Alan Baade.

I want to thank my family, including relatives, for their love, support, and for always asking me how many years I have left in my PhD. Despite thousands of late nights at lab, I never felt like a lonely grad student because of my friends and family. They have made and kept me a happy soul. Finally, I want to thank the individual members of my family: my Dad, for being the model engineer and son I looked up to as I grew up; my Mom, for giving birth to me and my brother, and for being my biggest supporter and most constructive critic; and my brother, for motivating me by often saying to my robots: "He sucks, he does."

Abstract

Augmenting Robotic Manipulation through Natural Language

Albert Yu, PhD student
The University of Texas at Austin, 2026

SUPERVISORS: Raymond Mooney, Roberto Martín-Martín

Despite rapid advances in language and vision models, current robots still lag far behind human physical capabilities due to the relative scarcity of real-world data compared to online text and images. How can we leverage abundant language data to advance robotic capabilities? Language provides semantic structure that facilitates the understanding of diverse data, improving sample efficiency in scarce data regimes. It also provides a natural communicative medium when interacting with and learning from humans.

To leverage the first benefit of language, we first take inspiration from how humans teach each other in video tutorials, through simultaneous video and language streams, to more efficiently teach robots new skills. We then show that language can bridge wide visual sim2real gaps, enabling robots to learn tasks with just a few real-world demonstrations by leveraging knowledge from imperfect simulation data. To leverage the second benefit of language, we explore how dialog can enable robots to solve complex manipulation tasks by communicating and collaborating with a wide distribution of human collaborators in the real-world. We develop a robotic framework that requests and proactively offers help through mixed-initiative, free-form dialog, enabling the robot to adapt to changing human preferences and strategically utilize each agent's physical capabilities. Finally, to accelerate how robots learn to operate

new household devices, we combine both benefits of language into a framework that leverages semantics from rich textual corpora, such as user manuals, to establish skill priors that are efficiently refined through active human dialog.

Overall, our thesis provides a foundation for leveraging the key benefits of language to improve the generalization capabilities of robotic manipulation policies to new tasks, domains, devices, and human collaborators.

Table of Contents

Chapter 1: Introduction

We envision robots as capable everyday helpers that can fetch items, cook, and clean. While millions of robots operate today with high precision in structured environments like factories, transferring robotic capabilities to unstructured (i.e., open-world) settings like households remains an open problem. Homes present an unbounded variety of object and scene configurations, often making classical robot control approaches brittle and unsafe. Because of this complexity, successful household robots have largely been limited to navigating robotic vacuums. Open-world robotic manipulation remains a formidable research challenge.

1.1 Scarcity of Robotic Data

For a robot to manipulate objects intelligently in unstructured settings, it must adapt to a dynamically changing world. Instead of pre-programmed motions, data becomes necessary for robots to learn patterns of acting intelligently, performing real-world tasks, and recovering from failures. While fields like natural language processing and computer vision leverage internet-scale datasets, robotics suffers from a scarcity of observation-to-action data.

Despite their potential in structured environments, the physical capabilities of manipulation robots in unstructured environments still lag behind humans, primarily due to the longstanding data scarcity in robotics. Data-hungry algorithms and architectures that serve well in data-abundant fields may not be the easiest path forward for robotics, given the expensiveness of collecting real-world data and training large models. How might we confront these issues?

1.2 How Language Can Contribute to Robotics

To act in the world, a robot must first perceive the world through sight and/or touch. After acting, the robot must observe the physical change it has caused to decide its next action. This perception-action loop was the focus for robotic control for several decades. However, as brought up in Section 1.1, paired perception and action data for robotics is scarce and expensive, limiting the complexity and generalizability of skills robots can learn. Other sources of data must be leveraged.

For several decades, natural language processing (NLP) was constrained to applications like translation, semantic parsing, and entity recognition. For robots to augment human physical capabilities, they must understand human need and intent, which are most easily specified through language. Robots that coexist with and serve humans must understand language, a modality not included in the traditional perception-action paradigm.

Furthermore, language brings additional benefits to robotics. Individual words are generally more abstract, conceptual, and meaningful than individual image pixels. While images provide precision to guide robot movements, language provides the ability for robots to extract semantic meaning from scarce data, associate rules-based behavior to specific visual inputs, reason about objects in complex scenes, and plan for long-horizon tasks.

Most importantly, freeform language data is plentiful and potent, as showcased by the post-2022 success of LLMs trained on the textual soup of the internet. This wealth of language-based knowledge has enabled incredible breakthroughs in automated programming, writing, and common-sense reasoning, but challenges remain in bringing the generalization power of LLMs to physical tasks.

1.3 Thesis Overview

Given these advantages of natural language, in this thesis, we focus on and investigate two core lines of work to leverage natural language to expand robotic capabilities. First, language is a compressed store of meaning through which we can more efficiently learn from scarce robotics data. Second, language is a rich communicative medium through which humans and robots can collaborate and adapt to each other by expressing their intentions, preferences, and capabilities.

After establishing the technical preliminaries in Chapter 2, our first line of work starts by exploring how semantic meaning captured in language can be exploited to enable efficient learning for few-shot generalization to both new tasks and new domains. In Chapter 3, we take inspiration from how humans teach each other in video tutorials, through simultaneous video and language streams, to more efficiently teach robots new skills. We show that providing natural language instructions along with a single visual demonstration greatly improves sample efficiency when learning novel tasks compared to previous methods that teach robots with only one modality. However, sometimes it is expensive for humans to teach robots through multiple simultaneous modalities in the real-world. Researchers have tried to leverage cheap, abundant simulation data to train robots, but the sim2real domain gap often degrades performance of simulation-trained policies. In Chapter 4, we show that language can help shape the learning of visual representations to better bridge large visual differences between sim and real, even when the sim2real gap is large and involves hard-to-simulate deformable objects. This enables robots to generalize to real-world domains with just a few real-world demonstrations.

Along the second line of work, in Chapter 5, we explore how language can enable smooth human-robot collaboration to accomplish complex manipulation tasks that the robot cannot easily learn during deployment. We argue that mixed-initiative dialog, which enables either the robot or human to start a conversational thread, can greatly facilitate human-robot collaboration, improving the adaptability of each agent

to the other's capabilities. We develop a robotic framework capable of collaborating with a wide range of real human participants through bidirectional, mixed-initiative, free-form dialog. Our method outperforms an LLM baseline by 50 percentage points on long-horizon mobile manipulation tasks and was preferred by more than 75% of the 18 participants.

Finally, Chapter 6 unifies these two lines of work with a framework that extracts manipulation skills from device user manuals. It first leverages the first benefit of language as a store of semantic meaning to initialize skill and policy priors on new devices. However, text alone lacks the high-precision geometric and force information required for physical control. To acquire this missing information, our framework leverages the second benefit of language as a communicative medium by actively querying humans for on-demand feedback through natural dialog. Ultimately, this combined approach transfers to new devices within a given category far more sample-efficiently than existing continual learning baselines.

Looking beyond our contributions, Chapter 7 outlines promising future directions. We discuss how to further utilize language to accelerate reinforcement and transfer learning, tap into richer forms of language beyond instructions, dialog, and scene descriptions, and enable smoother, simultaneous human-robot interaction and collaboration. We conclude in Chapter 8 with the expectation that future work along these directions, extending the frameworks pioneered in this thesis, promises to enable ever more capable and helpful robots that can in turn augment human productivity on everyday tasks.

Chapter 2: Background and Related Work

For robots to perform useful manipulation tasks in unstructured environments, we established in Section 1.1 that robots need to learn intelligent patterns of behavior from data. There are two main paradigms for doing so: imitation learning and reinforcement learning.

2.1 Imitation Learning

In imitation learning (Pomerleau, 1988; Hussein et al., 2017), also known as behavioral cloning (BC), we assume access to an expert teacher that provides demonstrations of behaviors that the robot should imitate, conditioned on the observation o_t (i.e., RGB image and/or robot xyz end-effector position). Each expert demonstration (usually a human teleoperation of the robot) is collected as a sequence of observation-action tuples indexed by the timestep t: $[(o_t, a_t), ...]$, where a_t represents the action taken from observation o_t (i.e., the commanded change in xyz position of the robot's end effector).

The goal of imitation learning is to train a policy $\pi_\theta : o_t \mapsto a_t$ that maps observations to actions, where $o_t \in \mathbb{R}^{d_o}$ and $a_t \in \mathbb{R}^{d_a}$. More commonly, $\pi_\theta : o_t \mapsto P(a_t|o_t)$, a probability distribution over possible actions. We want to learn the parameters θ of π, namely the weights of the neural network representing π, such that the error ε is minimized between the policy-predicted action $\hat{a}_t$, and the expert demonstration action a_t: $\varepsilon = ||a_t - \hat{a}_t||_2^2$, where $\hat{a}_t \sim \pi_\theta(\cdot|o_t)$. There are two common losses used to minimize ε: ℓ_2 or log-likelihood. Let $\mathcal{D}$ be the set of demonstrations in our dataset, each of which is a trajectory τ_i.

When using ℓ_2 loss, the goal is to find the optimal policy parameters:

$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_{\tau_i \in \mathcal{D}} \sum_{(o_{t,i}, a_{t,i}) \sim \tau_i} ||a_{t,i} - \pi_\theta(\cdot|o_{t,i})||_2^2 \tag{2.1}$$

where the outer sum is over all expert trajectories in our dataset, and the inner sum is over all observation-action transitions in our trajectory.

When using log-likelihood loss, the goal is to find optimal policy parameters:

$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_{\tau_i \in \mathcal{D}} \sum_{(o_{t,i}, a_{t,i}) \sim \tau_i} -\log \pi_\theta(a_{t,i}|o_{t,i}) \tag{2.2}$$

Essentially, this seeks to maximize the product of the policy's probabilities of predicting all actions $a_{t,i}$ in expert demo τ_i. This is the same optimization problem as maximizing the sum of log probabilities of the policy, which is equivalent to minimizing the negative sum of log probabilities.

By following either of these objectives, we train a policy π_θ to follow the expert's actions in observations seen by the expert and rely on the interpolative power of neural networks to be robust to small deviations in the observation from the training distribution.

2.2 Reinforcement Learning

Often, expert demonstrations are hard to obtain. For instance, human expert-level teleoperation is difficult on multi-legged robots. Additionally, imitation learning yields policies upperbounded by the expert's performance. These considerations may make reinforcement learning (RL) better suited for training robots. Instead of expert demonstrations, RL requires a reward function (i.e., performance metric) that measures how good or bad a state-action pair is toward achieving the task (Sutton and Barto, 2018).

RL operates in a Markov-Decision Process (MDP), defined as a tuple $\mathcal{M} = (S, A, R, S', \mathcal{P})$, where S is the set of start states from where actions A can be taken, S' is the set of next states the agent lands in after taking an action, $\mathcal{P} : S \times A \to P(S')$ is the conditional probability distribution over next states after taking action $a \in A$ from state $s \in S$. At each timestep, the agent receives scalar reward $R(s, a)$, where R is a function mapping states and actions to a scalar.

The policy randomly explores the state-action space and finds a sequence of actions to maximize the sum of discounted rewards:

$$\mathbb{E}_{(s_t,a_t,s_{t+1})}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\right] \tag{2.3}$$

Let $\pi(\cdot|s_t)$ represent the action predicted by the policy from state s_t. We define two functions that correspond to the "goodness" of a state when acting under a policy π. The first is the value function, or the expected sum of discounted rewards that the policy collects from state s_0 to termination.

$$V^\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R\big(s_t,\pi(\cdot|s_t)\big)\right] \tag{2.4}$$

$$= R(s_0,\pi(\cdot|s_0)) + \gamma\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R\big(s_{t+1},\pi(\cdot|s_{t+1})\big)\right] \tag{2.5}$$

$$= R\big(s_0,\pi(\cdot|s_0)\big) + \gamma V^\pi(s_1) \tag{2.6}$$

The second is the quality function, or the expected reward from state s_0, taking action a_0, with all future actions dictated by policy π.

$$Q(s_0,a_0) = R(s_0,a_0) + \gamma Q^\pi(s_1,a_1) \tag{2.7}$$

Both V and Q can be defined recursively, as seen above, which is key to learning these functions using one of many RL algorithms for the policy to learn to perform optimal actions. We refer the reader to surveys detailing different classes of RL algorithms (Arulkumaran et al., 2017; Ghasemi et al., 2024).

Multiple works have explored using language to shape the reward function R for an RL agent (Nair et al., 2021b; Goyal et al., 2019, 2020; Fan et al., 2022; Ma et al., 2023, 2024a). Some researchers have specifically explored using language models (LLMs) to generate code to tune a reward function when training robotic RL policies (Ma et al., 2024b; Yu et al., 2023a).

2.3 Multitask Learning

So far, we have discussed training our policy π to perform a specific task, either through imitation learning or RL. However, in unstructured environments, we want robots to be able to perform a wide range of tasks. Under the current tools discussed, we would need to train a single-task policy from scratch for each new task we want the robot to perform. This is unscalable and prevents us from leveraging data for one task in learning a related second task. To overcome these problems, multitask learning (Wilson et al., 2007a; Taylor and Stone, 2009) is necessary.

Earlier in the section, our policy was conditioned on only the state or observation $\pi : S \rightarrow P(A)$. However, a multitask policy π may need to output different actions from the exact same state, depending on what the task is. Thus, we additionally condition π on a task conditioning vector C, so $\pi : S \times C \rightarrow P(A)$. C can be as simple as a one-hot vector, goal image embedding, or natural language instruction embedding.

Work in multi-task learning suggests that training on a wide range of tasks, instead of the single target task, helps the robot learn shared perceptual representations across the different tasks, improving generalization (Kalashnikov et al., 2021; Yu et al., 2019). The most straightforward way to condition multi-task policies is through one-hot vectors (Ebert et al., 2021; Kalashnikov et al., 2021; Walke et al., 2022; Yu et al., 2021b). Multi-task robotic policies have also been studied in other settings and contexts, such as hierarchical goal-conditioned policies (Gupta et al., 2022a), probabilistic modeling techniques (Wilson et al., 2007b), distillation and transfer learning (Parisotto et al., 2015; Teh et al., 2017; Xu et al., 2020; Rusu et al., 2015), data sharing (Espeholt et al., 2018; Hessel et al., 2019), gradient-based techniques (Yu et al., 2020), policy modularization (Andreas et al., 2017; Devin et al., 2017) and task modularization (Yang et al., 2020). However, one-hot conditioning fails to leverage similarity information between related tasks. We propose an approach to address this issue in Section 3.

2.4 Vision and Language

Researchers have explored how to ground natural language in visual concepts or objects by developing models that can understand both visual and textual modalities. Vision-language research first found success with problems like visual question-answering (VQA; Agrawal et al. (2015); Marino et al. (2019)), image captioning (Socher et al., 2014; Kiros et al., 2014; Mao et al., 2014; Vinyals et al., 2014), and video captioning (Venugopalan et al., 2015; Yao et al., 2015) in the mid 2010s. The first attempts at training models for the inverse problems of text-to-image generation (Mansimov et al., 2015; Reed et al., 2016) and text-to-video generation (Pan et al., 2017; Li et al., 2017) came shortly after.

Transformers arose as a neural architecture for pure language tasks but soon unified the processing of both images and text with the first vision-language transformers (Lu et al., 2019; Sun et al., 2019; Li et al., 2019; Tan and Bansal, 2019; Su et al., 2019; Chen et al., 2019), some of which have been applied to robotic control (Shridhar et al., 2021; Zeng et al., 2020; Shridhar et al., 2022; Cui et al., 2022; Zeng et al., 2022; Brohan et al., 2023, 2022). In addition to architectures, researchers have also examined learning better vision-language joint representations (Radford et al., 2021; Zhai et al., 2021; Zhu et al., 2023) to improve robotic control (Nair et al., 2022; Shridhar et al., 2021, 2022).

These prior works in vision-language research form the technical basis of enabling language-conditioned, vision-based robotics in the real-world. However, visual representations learned from these prior approaches yield robotic policies that largely fail to generalize to large domain shifts. We address this problem in Section 4.

2.5 Natural Language for Robot Task Specification

As mentioned in Section 2.3, multitask policies take in context C, which can be in the form of a natural language embedding of the task instruction, making it a language-conditioned multitask policy (Jang et al., 2021; Lynch and Sermanet,

2021; Mees et al., 2021, 2022; Shao et al., 2020; Sodhani et al., 2021; Silva et al., 2021; Karamcheti et al., 2021; Garg et al., 2022). Pretrained language embedding spaces normally preserve a notion of semantic similarity through distance—that is, two strings similar in meaning will be encoded into two language embeddings close in vector-space distance. This means that language-conditioned multitask policies have two benefits: robustness (slight rewordings of the language instruction do not meaningfully change the language embedding) and generalizability to new tasks (a new, related task will have a language embedding close to those of semantically related training tasks). However, teaching robots only through language can lead to a lot of ambiguities, which we address in Section 3.

LLMs have also been used as task planners (Huang et al., 2022; Ahn et al., 2022; Chen et al., 2022; Raman et al., 2023; Choi et al., 2025; Luo et al., 2023), as code generators that dictate a robotic policy's behavior (Liang et al., 2022; Li et al., 2024a; Huang et al., 2023), and as part of a hierarchical policy where the higher level produces language and the lower level produces fine-grained robot actions (Shi et al., 2025, 2024; Belkhale et al., 2024).

2.6 Natural Language for Multi-Agent and Human-Robot Interaction

While language can serve as a one-way communication medium to teach or instruct robots, it can also facilitate bidirectional dialog to enable Human-Robot Interaction (HRI). Some systems integrate LLMs as task planners or delegators (Wang et al., 2024; Mandi et al., 2023; Feng et al., 2024) for tasks like real-world cooking (Wang et al., 2024) and object sorting (Mandi et al., 2023), where task delegations are communicated through dialog. Other systems implement a leader-follower paradigm in simulated worlds, where the leader instructs the follower in natural language (Suhr et al., 2022; Kojima et al., 2021; Team et al., 2022; Gao et al., 2023).

Another line of work empowers the robot to request assistance (Bennetot et al.,

2020; Knepper et al., 2013; Veloso et al., 2015) or inform humans about their observations (Chen et al., 2010; Mutlu et al., 2006; Cascianelli et al., 2018). Researchers have also studied multi-agent communication for ad-hoc teamwork in a simplified symbolic language (Barrett et al., 2014) or in natural lanugage, where one agent asks while the other responds (Macke, 2023).

We argue that these prior works do not exploit the full flexibility of language as a communicative medium to the extent needed for adaptable and versatile human-robot collaboration. To this end, we present a new human-robot dialog framework in Section 5 that enables both agents to smoothly seize and relinquish initiative in long-horizon collaborative tasks, allowing agents to mutually adapt to each other's capabilities to improve task success and user satisfaction.

2.7 Active Learning with Language

In imitation learning (Sec. 2.1), a human usually provides a set of demonstrations upfront to the robot to teach it a new task, and the robot processes these demonstrations on its own to learn a policy. However, if the human is still available while a robot is learning, the robot may learn faster by querying the human for specific information—a paradigm known as active learning. In the robot learning and language grounding domains, active learning is most commonly used to learn new concepts, such as plan steps (Nyga et al., 2018), prepositional phrases (Kulick et al., 2013), and unknown skills and task structures (Gu et al., 2026).

In learning new concepts, agents often struggle to ground novel linguistic concepts due to out-of-vocabulary terms or semantic ambiguity. To mitigate this, prior work has studied how agents can formulate clarification questions within the active learning paradigm (Deits et al., 2013; Thomason et al., 2015; Ren et al., 2023). In other instances during human interaction, agents should occasionally sacrifice immediate task efficiency by asking off-topic questions to learn concepts for future use. Leveraging this insight, Padmakumar et al. (2018b); Thomason et al. (2017b) intro-

duce opportunistic active learning for interactive object retrieval. Finally, Thomason et al. (2018, 2019a); Padmakumar and Mooney (2021) combine both opportunistic active learning and clarification questions into a single active learning robotics system with a three-part objective: learn new concepts opportunistically, clarify ambiguities, and maximize task success. In Chapter 6, we perform a form of active learning, but unlike prior work, the robot does not query the human for skill feedback unless it self-detects a failure in its skill execution.

2.8 Code-based Policy Representations

While end-to-end neural networks are the most common policy architectures for imitation learning (Sec. 2.1) and reinforcement learning (Sec. 2.2), a new policy representation has emerged in the last four years for language-conditioned robotic tasks. The dramatic improvement in code generation and semantic scene understanding capabilities of LLMs has driven the rise of code-based policy representations (Liang et al., 2022; Singh et al., 2022; Arenas et al., 2024), where an LLM processes a task instruction and creates functions in code that call a base robot API. The LLM uses in-context learning, where it infers patterns from previous program examples to produce higher-quality outputs (Brown et al., 2020). The generated code serves as an executable policy, guiding the robot's motions to complete the task. Code is a versatile policy representation for language-conditioned robotics and has been leveraged to encode 3D voxel-based cost functions for trajectory optimization (Huang et al., 2023), make decisions complying with user preferences (Wu et al., 2023a), reason through chain-of-thought as code execution predictions (Li et al., 2024b), and integrate with 3D position proposals for greater spatial precision (Mu et al., 2024).

In Chapter 5, we borrow the code-based policy representation for manipulation and apply it to the domain of collaborative manipulation through dialog, where the adaptiveness of code generated at each timestep enables the robot to strategically communicate and collaborate with a human. In Chapter 6, we explore how to leverage

large-scale textual corpora and human language feedback to improve the quality of code-based policies.

Chapter 3: Using Both Demonstrations and Language Instructions to Efficiently Learn Robotic Tasks

As outlined in Section 1.3, the first core line of this thesis investigates how language can serve as a semantic prior to generalize to new robotic tasks from scarce data. Based on work published at ICLR 2023 (Yu and Mooney, 2022), this chapter explores how combining this language-based semantic prior with visual data allows average users to teach household robots far more efficiently than using a single modality.

3.1 Introduction

Household robot deployment is heavily hindered by the fact that novice users cannot teach them new tasks with minimal time and effort. Recent work in multi-task learning suggests that training on a wide range of tasks, instead of the single target task, helps robots learn shared perceptual representations across the different tasks, improving generalization (Kalashnikov et al., 2021; Yu et al., 2019; Jang et al., 2021; Shridhar et al., 2021). We study the problem of how to more efficiently specify new tasks for multi-task robotic policies while also improving performance.

Humans often learn complex tasks through multiple concurrent modalities, such as simultaneous visual and linguistic (speech/captioning) streams of a video tutorial. One might reasonably expect robotic policies to also benefit from multi-modal task specification. However, previous work in multitask policies condition only on a single modality during evaluation: one-hot embeddings, language embeddings, or demonstration/goal-image embeddings. Each has limitations.

One-hot encodings for each task (Kalashnikov et al., 2021; Ebert et al., 2021) suffice for learning a repertoire of training tasks but perform very poorly on novel

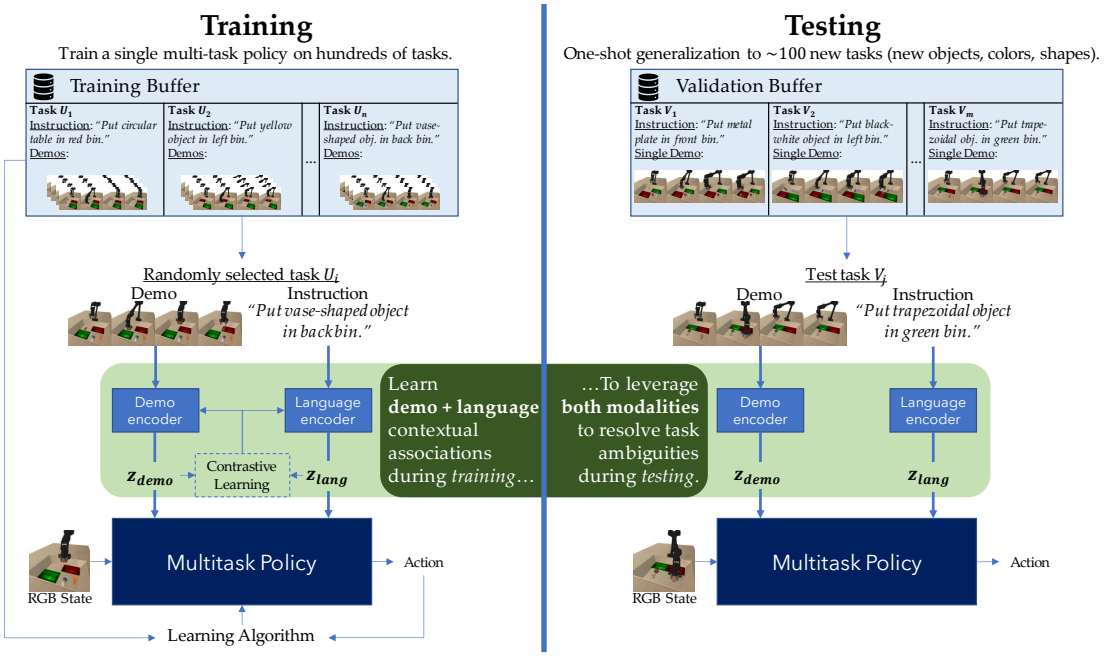


Figure 3.1: **DeL-TaCo Overview.** Unlike current multitask methods that condition on a single task specification modality, DeL-TaCo simultaneously conditions on both language and demonstrations during training and testing to resolve any ambiguities in either task specification modality, enabling better generalization to novel tasks and significantly reducing teacher effort for specifying new tasks.

tasks where the one-hot embedding is out of the training distribution, since one-hot embedding spaces do not leverage semantic similarity between tasks to more rapidly learn additional tasks. Conditioning policies on goal-images (Nair et al., 2017, 2018; Nasiriany et al., 2019) or training on video demonstrations (Smith et al., 2020; Young et al., 2020) often suffer from ambiguity, especially when there are large differences between the environment of the demonstration and the environment the robot is in, hindering the understanding of a demonstration's true intention. In language-conditioned policies (Blukis et al., 2018, 2019; Mees et al., 2021, 2022), issues of ambiguity are often even more pronounced, since humans specify similar tasks in very linguistically dissimilar ways and often speak at different levels of granularity, skipping over common-sense steps and details while bringing up other impertinent information. Grounding novel nouns and verbs not seen during training compounds these challenges.

We posit that in a broad category of tasks, current unimodal task representations are often too inefficient and ambiguous for novel task specification. In these tasks, current task-conditioning methods would need either a large number of diverse demonstrations to disambiguate the intended task, or a long, very detailed, fine-grained language instruction. Both are difficult for the average user to provide. We argue that conditioning the policy on *both* a demonstration *and* language not only ameliorates the ambiguity issues with language-only and demonstration-only specifications, but is *much easier and more cost-effective for the end-user to provide.*

We propose DeL-TaCo (Figure 3.1), a new task embedding scheme comprised of two component modalities that contextually complement each other: demonstrations of the target task and corresponding language descriptions. To our knowledge, this is the first work to demonstrate that specifying new tasks to robotic multi-task policies simultaneously with both demonstrations and language reduces teacher effort and improves generalization performance, two important characteristics of deployable household robots. With bimodal task embeddings, ambiguity is bidirectionally resolved: instructions disambiguate intent in demonstrations, and demonstrations help ground novel noun and verb tokens by conveying what to act on, and how. To learn several hundred tasks, we train a single imitation learning (IL) policy, conditioned on joint demonstration-language embeddings, to predict low-level continuous-space actions for a robot given image observations. Task encoders are trained jointly with the policy, making our model fully differentiable end-to-end.

To summarize, our main contributions are as follows: (1) We present a broad distribution of highly-randomized simulated robotic pick-and-place tasks where instructions or demonstrations alone are too ambiguous and inefficient at specifying novel tasks. (2) We propose a simple architecture, DeL-TaCo, for training and integrating demonstrations and language into joint task embeddings for few-shot novel task specification. This framework is flexible and learning algorithm-agnostic. (3) We show that DeL-TaCo significantly lowers teacher effort in novel task-specification

and improves generalization performance over previous unimodal task-conditioning methods.

3.2 Related Work

3.2.1 Multi-task Learning

The most straightforward way to condition multi-task policies is through one-hot vectors (Ebert et al., 2021; Kalashnikov et al., 2021; Walke et al., 2022; Yu et al., 2021b). We instead use embedding spaces that are shaped with pretrained language models so that semantically similar tasks are encoded in similar regions of the embedding space, which helps improve generalization. Multi-task robotic policies have also been studied in other settings and contexts that do not fall under the class of approaches we take in this paper, such as hierarchical goal-conditioned policies (Gupta et al., 2022a), probabilistic modeling techniques (Wilson et al., 2007b), distillation and transfer learning (Parisotto et al., 2015; Teh et al., 2017; Xu et al., 2020; Rusu et al., 2015), data sharing (Espeholt et al., 2018; Hessel et al., 2019), gradient-based techniques (Yu et al., 2020), policy modularization (Andreas et al., 2017; Devin et al., 2017) and task modularization (Yang et al., 2020).

3.2.2 Learning with Language and Demonstrations

3.2.2.1 Conditioning Multitask Policies on Language or Demonstrations

Our work largely tackles the same problem as BC-Z (Jang et al., 2021) of generalizing to novel tasks with multi-task learning. BC-Z trains a video demonstration encoder to predict the pretrained embeddings of corresponding language instructions, while jointly training a multi-task imitation learning policy conditioned on *either* the instruction *or* demonstration embeddings. Lynch and Sermanet (2021); Mees et al. (2021) learn a similar policy conditioned on either language or goal images. All of these approaches learn to map a demonstration or goal image to a similar embedding space as its corresponding language instruction. During training, Mees et al. (2022)

use both demonstrations and language to learn associations between demonstration embeddings and language-conditioned latent plans, but during evaluation, only use the language embedding to produce a latent plan. With a slightly different architecture, Shao et al. (2020) learn a policy that maps natural language verbs and initial observations to full trajectories by training a video classifier on a large dataset of annotated human videos.

While all of these prior approaches use both demonstrations and language during training, their policies are conditioned on *either* a language instruction *or* visual image/demonstration embedding during testing. By contrast, ours is conditioned on *both* demonstration *and* language embeddings during training and testing, which we show improves generalization performance and reduces human teacher effort on a broad category of tasks.

3.2.2.2 Pretrained Multi-modal Models for Multitask Policies

Another recent line of work leverages pretrained vision-language models to learn richer vision features for downstream policies. CLIPort (Shridhar et al., 2021) uses pre-trained CLIP (Radford et al., 2021) to learn robust Transporter-based (Zeng et al., 2020) robot policies. Our method resembles CLIPort, its 3-dimensional successor PerAct (Shridhar et al., 2022), and the previously mentioned multi-task policy methods in that we train on expert trajectories associated with language task descriptions. However, in CLIPort and PerAct, the policy is *only conditioned on language* during training and testing; demonstrations are used only as buffer data for imitation learning. Our method, in contrast, learns tasks during training or testing by using *both language and a demonstration* to condition the policy.

ZeST (Cui et al., 2022) and Socratic Models (Zeng et al., 2022) demonstrate that pretrained vision-language models encode valuable information for robotic goal selection and task specification. R3M (Nair et al., 2022) pretrains a ResNet (He et al., 2015) policy backbone on language-annotated videos from Ego4D (Grauman et al.,

2021) to boost downstream task performance. While our motivation is similar to ZeST in using a pretrained language model to leverage the structure of the pretrained embedding space, we assume access to both language and demonstrations for learning novel tasks and condition on task embeddings from both, which is unlike the ZeST and R3M problem settings where the policies are not directly task-conditioned.

3.2.3 Other Applications of Language for Robotics

3.2.3.1 Language-shaped state representations

On the MetaWorld multitask benchmark (Yu et al., 2019), a number of prior works have investigated using language instructions to learn better state representations. Sodhani et al. (2021) use language instruction embeddings to compute an attention-weighted context representation over a mixture of state encoders. Silva et al. (2021) learn a goal encoder that transforms language instruction embeddings to shape the state encoder representations.

3.2.3.2 Hierarchical Learning with Language

Our approach can be loosely framed as hierarchical learning, where we have two high-level task encoders that output language and demonstration embeddings, both of which the low-level policy is conditioned on to output actions. Prior work has used language instructions in hierarchical learning for shaping high-level plan vectors (Mees et al., 2022) or skill representations (Garg et al., 2022), which are then fed to a low-level policy to output the action. Karamcheti et al. (2021) use an autoencoder-based architecture to predict higher-dimensional robot actions from lower-dimensional controller actions and language instructions, where the language is fed into both the encoder and decoder. All of these prior approaches condition on a single high-level policy, whereas ours incorporates guidance from two high-level encoders for *both* demonstrations and language to learn novel tasks, giving the low-level policy access to certain information expressible only through their combination.

3.2.3.3 Language for Rewards and Planning

Language has also been used for reward shaping in RL (Nair et al., 2021b; Goyal et al., 2019, 2020). Pretrained language models have also been leveraged for their ability to propose plans in long-horizon tasks (Huang et al., 2022; Ahn et al., 2022; Chen et al., 2022). While we work with IL instead of RL and mainly deal with highly variable pick-and-place tasks, we do not use language for training reward functions or for planning, though our multi-modal task specification framework is compatible with these additional uses of language.

3.3 Problem Setting

3.3.1 Multi-task Imitation Learning

We define a set of n tasks $\{T_i\}_{i=1}^n$ and split them into training tasks U and test tasks V, where (U, V) is a bipartition of $\{T_i\}_{i=1}^n$. For each task T_i, we assume access to a set of m expert trajectories $\{\tau_{ij}\}_{j=1}^m$ and a single language description l_i. Given continuous state space $\mathcal{S}$, continuous action space $\mathcal{A}$, and task embedding space $\mathcal{Z}$, the goal is to train a Markovian policy $\pi : \mathcal{S} \times \mathcal{Z} \rightarrow \Pi(\mathcal{A})$ that maps the current state and task embeddings to a probability distribution over the continuous action space.

During training, we assume access to a buffer $\mathcal{D}_{\text{train}}$ of trajectories for only the tasks in U and their associated natural language descriptions. We define each trajectory as a fixed-length sequence of state-action pairs $\tau_{ij} = \left[\left(s_{0,j}^{(i)}, a_{0,j}^{(i)}\right), \left(s_{1,j}^{(i)}, a_{1,j}^{(i)}\right), ...\right]$, where j is the trajectory index for task $T_i \in U$ with task embedding z_i. We use behavioral cloning (BC) (Hussein et al., 2017; Pomerleau, 1988) to update the parameters of π to maximize the log probability of $\pi\left(a_{t,j}^{(i)} | s_{t,j}^{(i)}, z_i\right)$, though our framework is agnostic to the learning algorithm and would work for RL approaches as well.

During evaluation, we assume access to a buffer $\mathcal{D}_{\text{val}}$ of trajectories for only the tasks in V and their associated natural language descriptions. Unlike $\mathcal{D}_{\text{train}}$ where we have m demonstrations for each task, in $\mathcal{D}_{\text{val}}$ we have just a single demonstration for each task. For all test tasks $T_i \in V$, we rollout the policy for a fixed number

of timesteps by taking action $a_t \sim \pi(a|s_t, z_i)$. The z_i for all test tasks is computed beforehand and held constant throughout each test trajectory.

3.3.2 Task Encoder Networks

To obtain the task embedding z_i, we have two encoders (which are either trained jointly with policy π or frozen from a pretrained model): a *demonstration encoder*, $f_{demo} : \tau_{ij} \mapsto z_{demo,i}$ mapping trajectories of task T_i to demonstration embeddings, and a *language encoder*, $f_{lang} : l_i \mapsto z_{lang,i}$ mapping task instruction strings l_i to language embeddings. Previous work has explored using z_i as a one-hot task vector, language embedding $z_{lang,i}$, *or* goal image/demonstration embedding $z_{demo,i}$, but our approach DeL-TaCo uses task embedding $z_i = [z_{demo,i}, z_{lang,i}]$ based on *both* the instruction *and* demonstration embedding during training and testing to learn novel tasks.

3.4 Method

3.4.1 Architecture

Demonstration Encoder. The encoder f_{demo} is a CNN network trained from scratch. Following Jang et al. (2021), we input the demonstration as an array of $m \times n$ frames (in raster-scan order) from the trajectory for faster processing. We use $(m, n) = (1, 2)$ or $(2, 2)$ in our experiments, depending on whether f_{demo} is trained from scratch, or is a pretrained network that requires square-shaped images. The $m \times n$ image array consists of observations from the first timestep, the last timestep, and $mn-2$ other randomly selected timesteps from the trajectory, arranged in raster-scan order. This sparse sampling of frames suffices for our tasks because we do not experiment with long-horizon tasks, so including more frames did not improve performance.

Language Encoder. We freeze a pretrained miniLM (Wang et al., 2020) as the encoder f_{lang}, where $z_{lang,i}$ is simply the average of all miniLM-embedded tokens

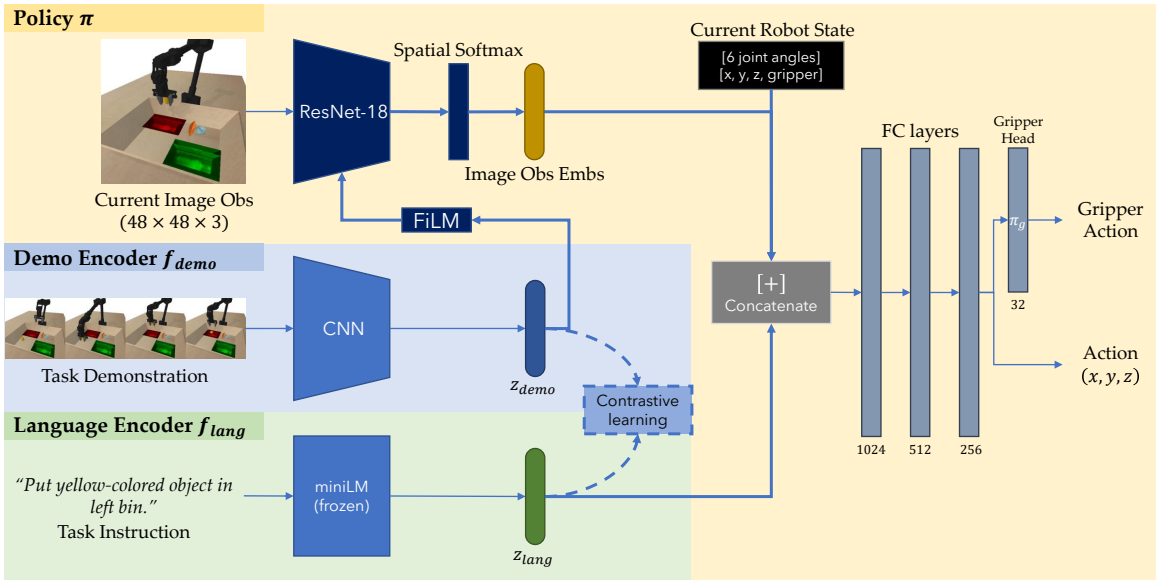


Figure 3.2: **Method Architecture.** DeL-TaCo uses three main networks: the policy π, a demonstration encoder f_{demo}, and a language encoder f_{lang}. During both training and testing, the policy is conditioned on the demonstration and language embeddings for the task.

in l_i, as we found this works better than taking the [CLS] token embedding.

Policy Network. We use a ResNet-18 (He et al., 2015) as the visual backbone for the policy π, followed by a spatial softmax layer (Finn et al., 2016) and fully connected layers.

Task Conditioning Architecture. BC-Z (Jang et al., 2021) inputs the task embedding into the ResNet backbone via FiLM (Perez et al., 2018) layers, which apply a learned affine transformation to the intermediate image representations after each residual block. BC-Z's task embeddings are either from demonstrations *or* language. Since our policy conditions on both, the main architectural decision was finding the best way to feed task embeddings from multiple modalities into the policy.

Empirically, a simple approach performed best. The demonstration embeddings z_{demo} are fed into the policy's ResNet backbone via FiLM, while the language task embeddings z_{lang} and robot proprioceptive state (6 joint angles, end-effector

xyz coordinates, and gripper open/close state) are concatenated to the output of the spatial softmax layer. Our full network architecture is shown in Figure 3.2, hyperparameters are in Sec. 3.4.3, and architectural ablations are in Table 3.10.

3.4.2 Training and Losses

The training procedure for DeL-TaCo is summarized in Algorithm 1. During each training iteration, we sample a size k subset of training tasks $M = \{T_{m_1}, ..., T_{m_k}\} \subset U$. Given a trajectory τ_{ij} for task T_{m_i} and corresponding natural language instruction l_i, we compute the demonstration embeddings $z_{emb,m_i} = f_{demo}(\tau_{ij})$ and language embeddings $z_{lang,m_i} = f_{lang}(l_i)$. We collect the embeddings of tasks in M in matrices $Z_{demo} = [z_{demo,m_1}, ..., z_{demo,m_k}]$ and $Z_{lang} = [z_{lang,m_1}, ..., z_{lang,m_k}]$.

To train the demonstration encoder, Jang et al. (2021) use a cosine distance loss to directly regress demonstration embeddings to their associated language embeddings. However, this causes demonstration embeddings to be essentially equivalent to the associated language embeddings for each task, undercutting the value of passing both to our policy. To preserve information unique to each modality while enabling the language and demonstration embedding spaces to shape each other, we train with a CLIP-style (Radford et al., 2021) contrastive loss for our demonstration encoder:

$$\mathcal{L}_{demo}(Z_{demo}, Z_{lang}) = CrossEntropy\left(\frac{1}{\beta} Z_{demo}^{\top} Z_{lang}, I\right) \tag{3.1}$$

where I is the identity matrix and β is a tuned temperature scalar. For some trajectory of state-[xyz action, gripper action] pairs $x_{t,i,j} = \left(s_{t,j}^{(i)}, \left[a_{t,j}^{(i)}, g_{t,j}^{(i)}\right]\right)$ extracted from expert demonstration τ_{ij} for task T_{m_i}, we use a weighted combination of standard BC log-likelihood loss for the xyz actions and MSE loss for the gripper actions. We abbreviate z_l and z_d for z_{lang} and z_{demo}:

$$\mathcal{L}_{\pi}(\tau_{ij}) = \sum_{x_{t,i,j} \in \tau_{ij}} -\log \pi\left(a_{t,j}^{(i)} | s_{t,j}^{(i)}, z_{d,m_i}, z_{l,m_i}\right) + \alpha_g \left\| g_{t,j}^{(i)} - \pi_g\left(s_{t,j}^{(i)}, z_{d,m_i}, z_{l,m_i}\right) \right\|_2 \tag{3.2}$$

π outputs a distribution over xyz actions, and the gripper head π_g of the policy network outputs a scalar $\in [0, 1]$ for the gripper action trained on a tuned $\alpha_g > 0$. Both f_{demo} and π networks are trained jointly with the following loss, for a tuned $\alpha_d > 0$:

$$\mathcal{L}(\pi, f_{demo}, f_{lang}) = \mathcal{L}_\pi(\tau_{ij}) + \alpha_d \mathcal{L}_{demo}(Z_{demo}, Z_{lang}) \tag{3.3}$$

Note $\mathcal{L}_\pi(\tau_{ij})$ is summed over all trajectories in the batch of training tasks M (double summation in Equation 3.2 ommitted for brevity). f_{lang} has no loss term because we freeze the pretrained language model and rely on its pretrained embedding space to shape the demonstration encoding space.

3.4.3 Detailed Architecture and Hyperparameters

See Figure 3.3 for a detailed diagram of the policy and demonstration encoder (for a higher-level overview, see Figure 3.2). For the policy backbone, we use a ResNet-18 architecture but made changes to the strides and number of channels to adapt the network to our small image size. Policy hyperparameters are shown in Table 3.1, Demo encoder hyperparameters are in and Table 3.2, and IL training hyperparameters are in Table 3.3. In each training iteration, we sample 16 random tasks from our training buffer and get 64 samples for each task, for a total batch size of 1024.

Table 3.3: Imitation learning hyperparameters.

Attribute	Value
Number of Tasks per Batch	16
Batch Size per Task	64
Learning Rate	3×10^{-4}
Gripper Loss Weight (α_g in $\mathcal{L}_\pi$)	30.0
Task Encoder Weight (α_d in $\mathcal{L}$)	10.0
Contrastive Learning Temperature (β in $\mathcal{L}_{demo}$)	0.1

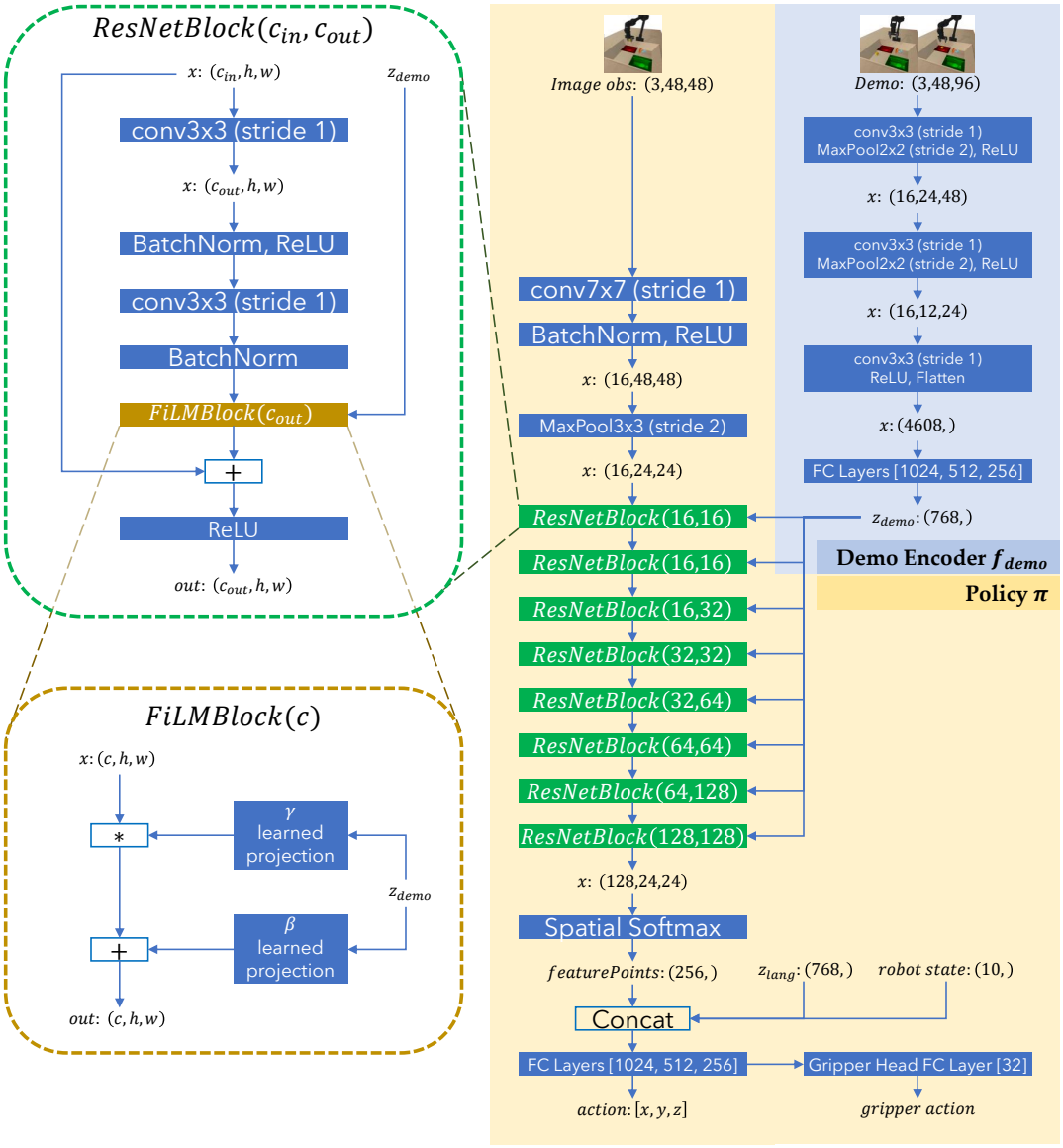


Figure 3.3: Architecture and Hyperparameter Details

3.4.4 Evaluation

During evaluation, we want the robot to perform some novel task $T_v \in V$. Recall that $T_v \notin U$, our set of training tasks. From our problem setup description in Sec. 3.3.1, we have access to a validation task buffer $\mathcal{D}_{\text{val}}$ with a single demonstration τ_v and a natural language instruction l_v for task T_v. We encode the demonstration

Table 3.1: Policy π hyperparameters.

Attribute	Value
Input Height	48
Input Width	48
Input Channels	3
Number of Kernels	[16, 32, 64, 128]
Kernel Sizes	[7, 3, 3, 3, 3]
Conv Strides	[1, 1, 1, 1, 1]
Maxpool Stride	2
Fully Connected Layers	[1024, 512, 256]
Hidden Activations	ReLU
FiLM input size	768
FiLM hidden layers	0
Spatial Softmax Temperature	1.0
Learning Rate	3×10^{-4}
Policy Action Distribution	Multivariate Isotropic Gaussian $\mathcal{N}(\mu, \sigma)$
Policy Outputs	(μ, σ)
Image Augmentation	Random Crops
Image Augmentation Padding	4

with f_{demo} and the language with f_{lang} and pass both task embeddings to the policy. Details are summarized in Algorithm 2.

Algorithm 1 DeL-TaCo: Training

Input: $\mathcal{D}_{\text{train}}$

while not done **do**
  $M \leftarrow k$ random train tasks from U
  Sample $X_i = \{\tau_{ij}\}_{j=0}^{b-1} \sim \mathcal{D}_{\text{train}}$
  $X \leftarrow \{X_i | T_i \in M\}$
  $L \leftarrow \{l_i | T_i \in M\}$ // Lang. instructions
  $Z_{demo} \leftarrow f_{demo}(X)$ // Demo encoder
  $Z_{lang} \leftarrow f_{lang}(L)$ // Language encoder
  Update π, f_{demo} on $\mathcal{L}(\pi, f_{demo}, f_{lang})$ // per Eqn. 3.3
end while

Algorithm 2 DeL-TaCo: Evaluation

Input: $\mathcal{D}_{\text{val}}$

for validation task T_v in V **do**
  Get 1 demo τ_v and language l_v from $\mathcal{D}_{\text{val}}$
  $z_{demo} \leftarrow f_{demo}(\tau_v)$ // Encode demo
  $z_{lang} \leftarrow f_{lang}(l_v)$ // Encode language
  for time $t = 0, ..., H-1$ **do**
    Take action $a_t \sim \pi(a | s_t, z_{demo}, z_{lang})$
  end for
end for

Table 3.2: f_{demo} CNN hyperparameters.

Attribute	Value
Demonstration frames	First and last timesteps
Demonstration image array size (m, n)	$(1, 2)$
Input Height ($m\cdot$ Image height)	48
Input Width ($n\cdot$ Image width)	96
Input Channels	3
Output Size	768
Kernel Sizes	[3, 3, 3]
Number of Kernels	[16, 16, 16]
Strides	[1, 1, 1]
Fully Connected Layers	[1024, 512, 256]
Hidden Activations	ReLU
Paddings	[1, 1, 1]
Pool Type	Max 2D
Pool Sizes	[2, 2, 1]
Pool Strides	[2, 2, 1]
Pool Paddings	[0, 0, 0]
Image Augmentation	Random Crops
Image Augmentation Padding	4

3.5 Experimental Setup

3.5.1 Environment

We develop a Pybullet (Coumans and Bai, 2007-2022) simulation environment with a WidowX 250 robot arm, 32 possible objects of diverse colors and shapes for manipulation, and 2 different containers. The action space is continuous, representing an (x, y, z) change in the robot's end effector position,

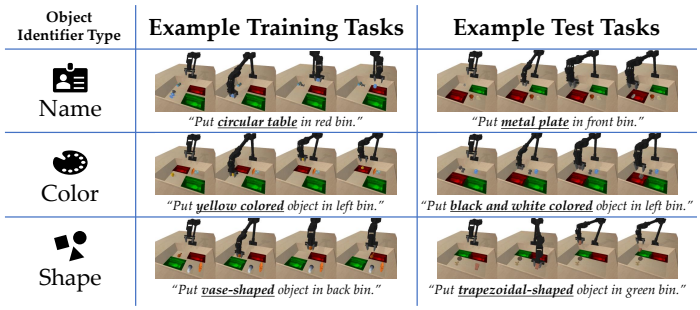


Figure 3.4: **Sample train and test tasks**, grouped by the object identifier types (underlined in each language instruction). All 6 container identifiers are seen in both training and testing.

plus the binary gripper state
(closed/opened). We subdivide the workspace into four quadrants. Two quadrants are randomly chosen to contain the two different containers, and three of the 32 possible objects are dropped at random locations in the remaining two quadrants. RGB image observations are size $48 \times 48 \times 3$ and fed into the policy. The input format of each demo for f_{demo} is an $m \times n$ array of images extracted from the trajectory.

3.5.2 Tasks

3.5.2.1 Task Objective

We design the following set of pick-and-place tasks where the objective is to grasp the target object and place it in the target container. Both the target object and container can be inferred from the demonstration and language instruction. In every task, the scene contains three visually distinct objects (of which exactly one of them is the target object) and two visually distinct containers (of which exactly one of them is the target container). Thus, a robotic policy that disregards both the task demonstration and instruction and picks any random object and places it into any random container would solve the task with 1-in-6 odds.

3.5.2.2 Template Language Instructions for Each Task

Figure 3.4 shows a selection of our training and testing tasks. Each task is specified through language with a single template-based instruction of the format "put [target object identifier] in [target container identifier]."

We make this environment more challenging by having task instructions refer to containers by either their color or quadrant position and objects by either their name, color, or shape. We use six different container identifiers in the task instructions to convey which container to drop the object in: red, green, front, back, left, and right. Thus, if the robot is provided a demonstration of grasping a cup and placing it in the red container in the front left quadrant, and it encounters an initialization

with the containers in different locations, it does not know whether to place the cup in the red, front, or left container. This ambiguity can only be resolved with the language instruction. Conversely, aspects of the task, such as which object to grasp, are most clearly expressed through the demonstration rather than the instruction, since for novel tasks, the language instruction contains object identifiers unfamiliar to the policy. The task instructions also refer to the objects through different types of identifiers: their unique names (32 strings such as "fountain vase"), color (8 strings such as "black-and-white colored object"), or shape (10 strings such as "trapezoidal prism shaped object").

The multiple identifiers help simulate ambiguity that arises from informal human instructions, where different humans may refer to the same object or container through different attributes, enabling demonstrations and instructions to complement each other when the robot learns a new task. In total, there are 50 target object identifiers ($32 + 8 + 10$) and 6 target container identifiers, giving us 300 pick-and-place tasks. We train and evaluate on a bipartition of these 300 tasks. See Sec. 3.5.2.3 for a list of all our train and test tasks.

Recall that we use the following template as the language instruction for each task: "Put [target object identifier string] in [target container identifier string]." For each object identifier, we build a string referring to the target obj in a specific format shown in Table 3.4.

Table 3.4: **Object Identifier String Format for each Object Identifier Type.**

Object Identifier Type	Target Object Identifier String
Name	"[object color] colored, [object shape] shaped [object name]"
Color	"[object color] colored object"
Shape	"[object shape] shaped object"

Example task instructions (with target object identifier and target container strings underlined):

- Task 4: "Put black and white colored, chalice shaped smushed dumbbell in green bin."
- Task 292: "Put cup shaped object in right bin."

3.5.2.3 List of Tasks

All 300 tasks are shown in Table 3.5, by object identifier (rows) and container identifier (columns). The colors denote groups of tasks which guide our train and test task splits, and the cell numbers denote the task indices.

- Scenario A (novel objects, colors, and shapes) trains on all gray tasks and tests on yellow, blue, and green tasks.
- Scenario B (novel colors and shapes) trains on all gray and yellow tasks and tests on blue and green tasks.

Table 3.5: Object and Container Identifiers

Object Identifier Type	Object Identifier	Container Identifier					
		green	red	front	back	left	right
name	conic cup	0	50	100	150	200	250
	fountain vase	1	51	101	151	201	251
	circular table	2	52	102	152	202	252
	hex deep bowl	3	53	103	153	203	253
	smushed dumbbell	4	54	104	154	204	254
	square prism bin	5	55	105	155	205	255
	narrow tray	6	56	106	156	206	256
	columnade top	7	57	107	157	207	257
	stalagcite chunk	8	58	108	158	208	258
	bongo drum bowl	9	59	109	159	209	259
	pacifier vase	10	60	110	160	210	260
	beehive funnel	11	61	111	161	211	261
	crooked lid trash can	12	62	112	162	212	262
	toilet bowl	13	63	113	163	213	263
	pepsi bottle	14	64	114	164	214	264
	tongue chair	15	65	115	165	215	265
	modern canoe	16	66	116	166	216	266
	pear ringed vase	17	67	117	167	217	267
	short handle cup	18	68	118	168	218	268
	bullet vase	19	69	119	169	219	269
	glass half gallon	20	70	120	170	220	270
	flat bottom sack vase	21	71	121	171	221	271
	trapezoidal bin	22	72	122	172	222	272
	vintage canoe	23	73	123	173	223	273
	bathtub	24	74	124	174	224	274
	flowery half donut	25	75	125	175	225	275
	t cup	26	76	126	176	226	276
	cookie circular lidless tin	27	77	127	177	227	277
	box sofa	28	78	128	178	228	278
	two layered lampshade	29	79	129	179	229	279
	conic bin	30	80	130	180	230	280
	jar	31	81	131	181	231	281
color	black and white	32	82	132	182	232	282
	brown	33	83	133	183	233	283
	blue	34	84	134	184	234	284
	gray	35	85	135	185	235	285
	white	36	86	136	186	236	286
	red	37	87	137	187	237	287
	orange	38	88	138	188	238	288
	yellow	39	89	139	189	239	289
shape	vase	40	90	140	190	240	290
	chalice	41	91	141	191	241	291
	freeform	42	92	142	192	242	292
	bottle	43	93	143	193	243	293
	canoe	44	94	144	194	244	294
	cup	45	95	145	195	245	295
	bowl	46	96	146	196	246	296
	trapezoidal prism	47	97	147	197	247	297
	cylinder	48	98	148	198	248	298
	round hole	49	99	149	199	249	299

Figure 3.5: **Train-Test Object Split**. Objects are shown in raster-scan task-index order, so the object in the second row from top, second column from left, is the "bongo drum bowl", which is associated with task index 9.

3.5.2.4 Train and Test Split Visualizations

We visually show our train-test splits on objects (Figure 3.5), colors (Figure 3.6), and shapes (Figure 3.7).

3.5.3 Success Metric.

In calculating the success rate, a successful trajectory is defined as one that (1) picks up the correct object *and* (2) places it in the correct container. All metrics are averaged over three seeds. See Appendix A.1 for details.

3.5.4 Data.

Using a scripted policy (details in Appendix A.2), we collect 200 successful demonstrations for each training task, and a single successful demonstration for each

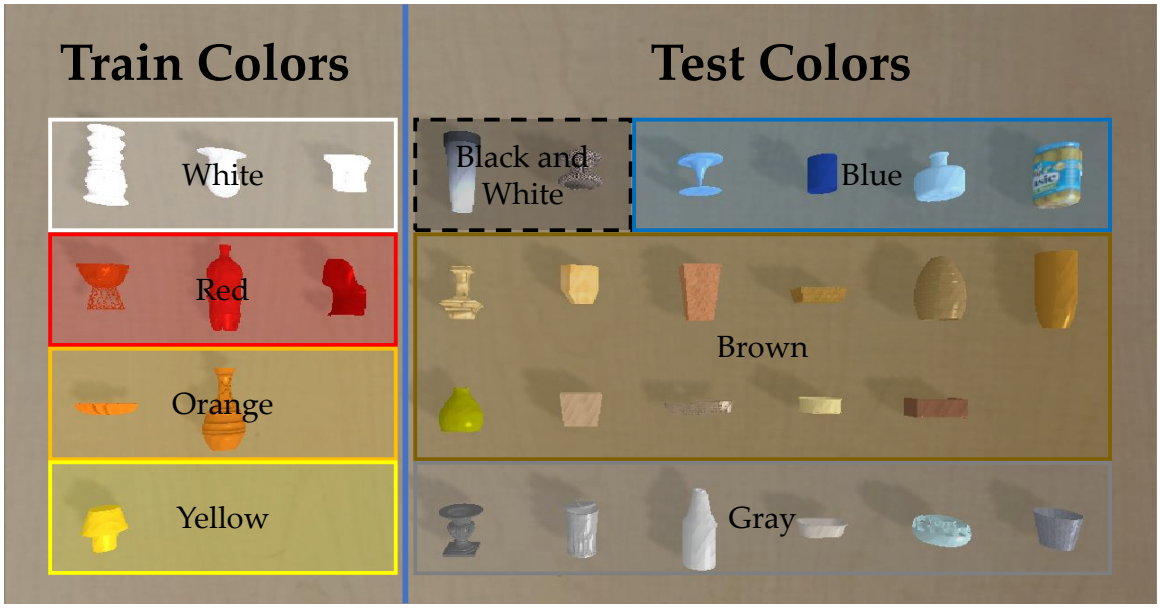


Figure 3.6: **Train-Test Color Split.**

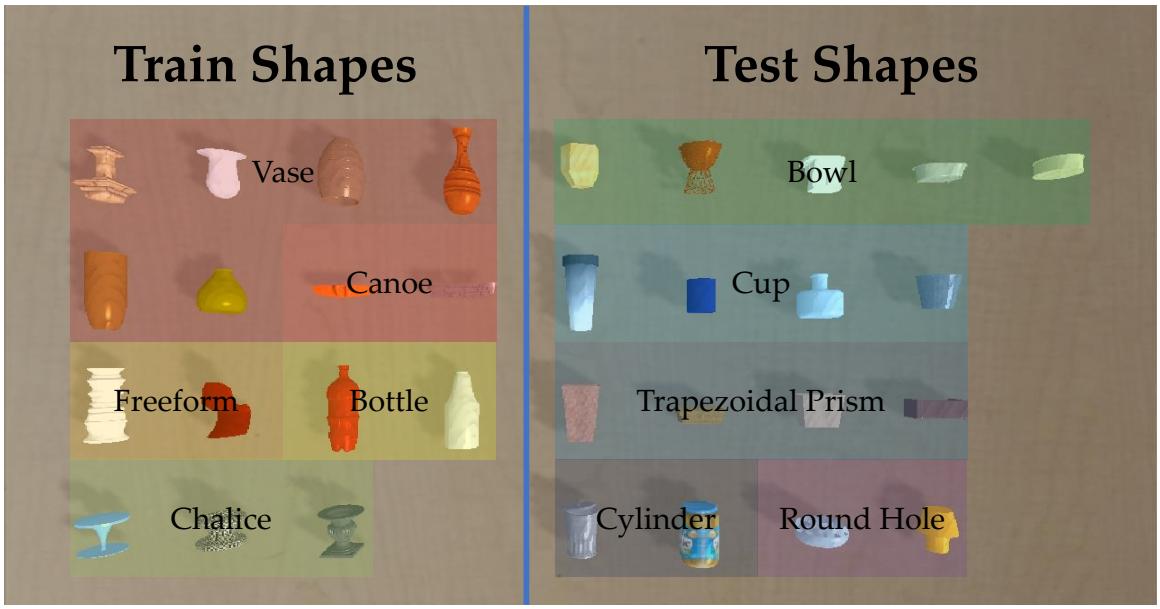


Figure 3.7: **Train-Test Shape Split.**

test task. All demonstrations are 30 timesteps long. Depending on our experimental scenario (see Section 3.6.1), we train on 65% to 80% of the 300 tasks, so our training buffer contains roughly 40,000-50,000 trajectories.

3.5.5 Human Language Instruction Paraphrases.

To move toward robots that can work alongside and assist humans in the real world, a rigorous language-conditioned robotic manipulation benchmark should contain not only template-based task instructions, but also natural language instructions from humans. To gather this data, we create a task (HIT) on Amazon Mechanical Turk for humans to paraphrase our template instructions while having access to example video rollouts of the robot successfully completing the the task. A CSV containing 5 paraphrases for each of the 300 tasks can be found on our project website (`https://deltaco-robot.github.io/files/all_human_data.csv`). See Appendix A.3 for details of our human language data collection process.

3.6 Experimental Analysis

We empirically investigate the following research questions: **(RQ1)** Does there exist a distribution of tasks that are more clearly specified with both language and demonstrations rather than either alone? **(RQ2)** Does conditioning on both language instructions and video demonstrations with DeL-TaCo improve generalization performance on novel tasks? **(RQ3)** If so, how much teacher effort is reduced by specifying a new task with both language and demonstrations than with either modality alone?

3.6.1 Generalization Performance on Novel Tasks

To test generalization, we run experiments under two scenarios: (A) generalization to novel objects, colors and shapes, and (B) generalization to only novel colors and shapes.

Table 3.6: Evaluation on Novel Objects, Colors, and Shapes. (p) = pretrained.

Demo Encoder	Language Encoder	Task Conditioning	Success Rate ± SD (%)
–	–	One-hot (lower bound)	4.9 ± 1.7
–	–	One-hot Oracle (upper bound)	69.3 ± 7.4
CLIP (p)		Language-only	14.4 ± 2.4
		Demo-only	3.5 ± 1.0
		DeL-TaCo (ours)	$\mathbf{15.8 \pm 2.2}$
–	miniLM (p)	Language-only	17.1 ± 2.2
CNN	–	Demo-only	20.8 ± 2.4
CNN	miniLM (p)	**DeL-TaCo (ours)**	$\mathbf{25.8 \pm 3.4}$
CNN	–	BC-Z (Jang et al., 2021); Demo-only	6.7 ± 2.3
CNN	miniLM (p)	MCIL (Lynch and Sermanet, 2021); Demo-only + Language-only	7.5 ± 1.2

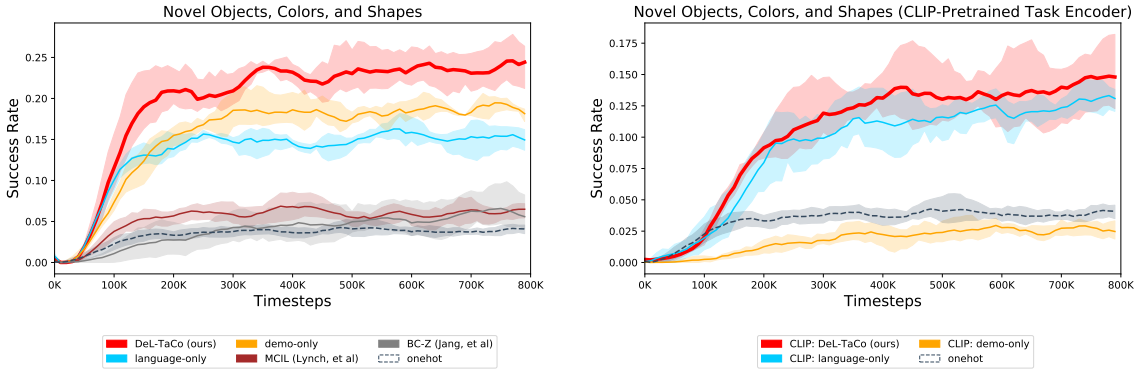


Figure 3.8: Learning curves for the results in Table 3.6, where all methods are evaluated on novel objects, colors, and shapes. **Left:** f_{demo} is a trained-from-scratch CNN and f_{lang} is pretrained miniLM. **Right:** f_{demo} and f_{lang} are from pretrained CLIP. The same one-hot lower-bound (dotted) is shown in both the left and right plots.

3.6.1.1 Scenario A: Novel Objects, Colors, and Shapes

Table 3.6 (plots in Figure 3.8) shows our results in experimental scenario A, where we train on 24/32 objects, 4/8 colors, and 5/10 shapes (a total of 198 training tasks) and evaluate on the remaining 102 tasks. The overall success rates show much room for improvement on this challenging benchmark for a number of reasons. The robot must not only know how to pick-and-place the 8/32 objects it has never seen during training, but must also understand novel instructions that refer to these objects by either their name, color, or shape. Additionally, training a multitask policy to perform well on hundreds of tasks remains an open question with current multitask robot learning algorithms.

We lower-bound the performance of our task conditioning methods by first running a one-hot conditioned policy, with the expectation that it performs worse than conditioning on language and/or demonstrations for the reasons mentioned in Section 3.1. As an upper-bound, we directly train a one-hot oracle on only the 102 evaluation tasks and evaluate on those same tasks. No other method in the table is trained on any evaluation tasks. (For consistency, the one-hot oracle is trained on the same total number of trajectories as the other methods.)

Next, we examine the performance of policies conditioned with only language, with only one demonstration, and with both (DeL-TaCo). The language-only policies do not involve training f_{demo}, and only the language instruction embeddings are fed into the policy via FiLM during training and testing. The demo-only policies train f_{demo} as shown in Algorithm 1, but during training and testing, only the demonstration embedding z_{demo} is passed into the policy via FiLM. DeL-TaCo (ours) conditions on *both* demonstration *and* language during training and testing as shown in Algorithms 1 and 2.

When using pretrained MiniLM as f_{lang} and a lightweight CNN for f_{demo}, DeL-TaCo achieves the highest performance, increasing the success rate of the second-best conditioning method, demo-only, from 20.8% to 25.8%. Both methods using demonstration embeddings outperform the language-conditioned policy perhaps because a visual demonstration is important in conveying the nature of the chosen object and how the robot should manipulate it. Note that both the demo-only and DeL-TaCo policies train the f_{demo} CNN from scratch without any pretraining, so they must learn to ground object and container identifiers from training demonstrations alone.

We also compare to our re-implementation of prior methods, keeping the architecture and hyperparameters the same across all methods. BC-Z (Jang et al., 2021) performs worse than our approaches because its demo encoder is trained to directly regress z_{demo} to z_{lang}, hindering it from performing better than solely using z_{lang} during testing. MCIL (Lynch and Sermanet, 2021), which trains separate en-

coders for each task embedding modality and averages the imitation learning losses over the different encoders, also performs worse than DeL-TaCo because without any task encoder loss term, learning a well-shaped task embedding space is more difficult, hurting generalization performance.

Finally, to evaluate the effect of pretraining, we use pretrained CLIP (Radford et al., 2021) as the task encoder (with its language transformer as f_{lang} and vision transformer as f_{demo}) and freeze it during training. The language-only policy performs significantly better than the video-only policy most likely because CLIP's visual transformer was trained mostly on real-world images and without further finetuning, does not know how to sufficiently differentiate between the simulation demonstrations of different tasks in our problem setting. Despite this, DeL-TaCo modestly outperforms conditioning on language-only or demo-only, demonstrating the value of our method even with frozen pretrained models.

3.6.1.2 Scenario B: Novel Colors and Shapes

In Table 3.7 (plots in Figure 3.9), we train on 32/32 objects, 4/8 colors, and 5/10 shapes, and evaluate on the rest—an easier setting as all objects were seen during training. Since evaluation tasks in this scenario only refer to objects by their color or shape, we up-sample the color and shape training tasks to be 50% of each training batch (such up-sampling was not done in scenario A).

We take the highest-performing f_{demo} and f_{lang} models in Table 3.6 and again compare conditioning on language, demonstrations, and both. The methods largely perform better in scenario B than A. The novel color and shape task demonstrations contain more ambiguity than the novel object demonstrations because the task with language instruction "put the blue object in the left bin" might have a demonstration where the robot manipulates the blue cup, but the test-time environment might contain a blue table instead. This added ambiguity likely explains the increased importance of language in supplementing the provided demonstration in DeL-TaCo.

Table 3.7: Evaluation on Novel Colors and Shapes. (p) = pretrained.

Demo Encoder	Language Encoder	Task Conditioning	Success Rate ± SD (%)
–	–	One-hot (lower bound)	8.4 ± 2.3
–	–	One-hot Oracle (upper bound)	61.7 ± 14.0
–	miniLM (p)	Language-only	25.4 ± 5.7
CNN	–	Demo-only	20.0 ± 4.0
CNN	miniLM (p)	**DeL-TaCo (ours)**	$\mathbf{31.6 \pm 5.2}$

Table 3.8: Value of Language. Evaluation on Novel Objects, Colors, and Shapes.

Task Conditioning	**Demo-only**					**DeL-TaCo (ours)**
# demos per test-task finetuned on	0	10	25	50	100	0
Success Rate (%)	20.8	23.4	24.6	<u>26.1</u>	32.9	<u>25.8</u>
± SD (%)	±2.4	±1.8	±2.5	±2.6	±2.5	±3.4

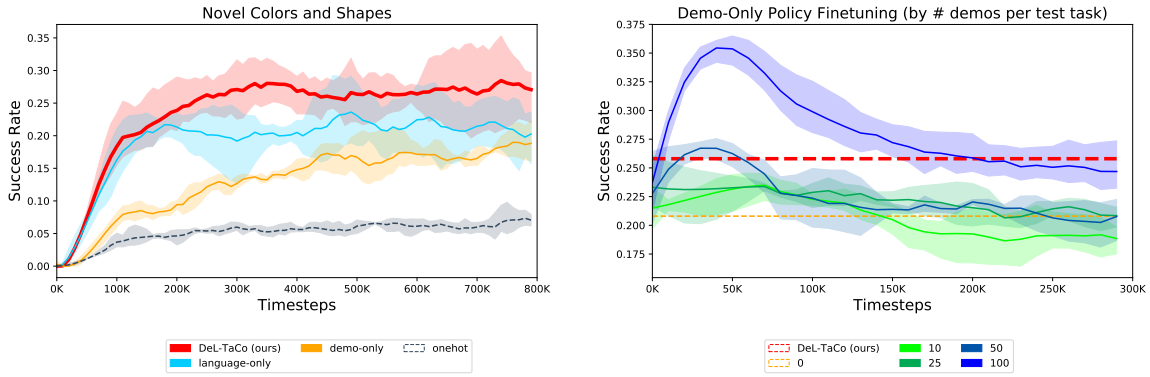


Figure 3.9: Learning curves for the results in Table 3.7 and 3.8. **Left:** Evaluation only on novel colors and shapes. **Right:** Evaluation on novel objects, colors, and shapes, using a trained-from-scratch CNN as f_{demo} and pretrained miniLM as f_{lang}. The performance of the demo-only policy and DeL-TaCo policy from Table 3.6 (also depicted in the left plot of Figure 3.8) are shown as lower and upper dotted lines, respectively. The solid lines indicate performance during 300k finetuning steps when given x demonstrations per test-task, where x is indicated in the legend.

Analysis. Overall, we see that on this wide range of tasks, language and demonstrations together do help disambiguate each other during task specification—answering **RQ1**; this leads to better generalization performance on novel tasks—answering **RQ2**.

Table 3.9: Ambiguity Experiments

Ambiguity Scheme	Demo Ambiguity	Language Ambiguity	Task Conditioning	Success Rate ± SD (%)
(i) (From Table 3.6)	–	None	Language-only	17.1 ± 2.2
	container	–	Demo-only	20.8 ± 2.4
	container	None	DeL-TaCo (ours)	$\mathbf{25.8 \pm 3.4}$
(ii)	–	object	Language-only	18.4 ± 2.5
	container	–	Demo-only	27.5 ± 4.3
	container	object	DeL-TaCo (ours)	$\mathbf{30.5 \pm 3.1}$

3.6.2 How many demonstrations is language worth?

To answer **RQ3**, we re-examine experimental scenario A (testing on novel objects, colors, and shapes) but now further finetune the demo-only policy on a variable number of test-task expert demonstrations. Results are shown in Table 3.8 (plots in Figure 3.9). The demo-only policy only starts to match and surpass DeL-TaCo (underlined) when it is finetuned on 50 demonstrations (underlined) *per evaluation task* (a total of around 5,000 demonstrations for all test tasks combined). This suggests that surprisingly, specifying a new task to DeL-TaCo with a single demonstration and language instruction performs as well as specifying a new task to a demo-only policy with a single demonstration *and finetuning it on 50 additional demonstrations of that task*. This showcases the immense value of language in supplementing demonstrations for novel task specification, significantly reducing the effort involved in teaching robots novel tasks over demonstration-only methods.

3.6.3 How effective is DeL-TaCo at resolving ambiguity in both modalities?

In Tables 3.6 and 3.7, we examined the effects of conditioning the multitask policy on a demonstration that is ambiguous about the target container, plus an unambiguous language instruction that clearly specifies both the target object and container for each pick-and-place task. We label this as ambiguity scheme (i).

To further examine the utility of multi-modal task conditioning, we experimented with a different ambiguity scheme (ii) in which the language instruction is *ambiguous* about the target *object* but *unambiguous* about the target *container*,

and the demonstration is *unambiguous* about the target *object* but *ambiguous* about the target *container*. This scheme allows us to test how well two task-conditioning modalities complement each other if each modality unambiguously conveys only a single aspect of the task.

To introduce ambiguity on the target objects, in scheme (ii), we place two identical objects on the scene in different parts of the tray, plus a third visually distinct distractor object. Exactly one of these two identical objects is the target object. The language instruction is ambiguous because it does not contain the positional identifier that describes which side of the tray the target object is at. However, the demonstration unambiguously conveys the target object by showing the robot picking up the target object from the correct part of the tray.

The results are shown in Table 3.9. By displaying two identical objects as we do in scheme (ii), we visually reveal that the target object is one of the two identical objects (and not the third, distinct object on the scene), causing scheme (ii) to have higher overall success rates than scheme (i). We see that the gap between DeL-TaCo and the demo-only policy slightly decreases from 5% in scheme (i) to 3% in scheme (ii) because the language instruction unambiguously specifies the task in scheme (i), complementing the demonstration which is ambiguous on both the target object and container, whereas in scheme (ii), both the language instruction and demonstration are ambiguous in one aspect (either in specifying the container or the object), narrowing the performance gap between DeL-TaCo and the demo-only policies.

3.6.4 Additional Module Comparisons and Ablations

In Table 3.10, we compare the performance of different language encoders, demonstration encoders, task encoder loss types, and task conditioning architectures in Experimental Scenario A to our best model from Table 3.6 (top row).

Language Encoder Model Comparison. In our main results, we use

Table 3.10: Ablations and Comparison Experiments. Evaluation on Novel Objects, Colors and Shapes. (p) = pretrained.

Demo Encoder	Language Encoder	Task Encoder Loss	Task Conditioning Arch.	Task Conditioning	Success Rate ± SD (%)
Best non-oracle result from Table 3.6					
CNN	miniLM (p)	Contrastive	FiLM (demo), Concat (lang)	DeL-TaCo (ours)	25.8 ± 3.4
Language Encoder Comparisons					
–	DistilRoBERTa (p)	–	FiLM	Language-only	15.1 ± 2.7
CNN	DistilRoBERTa (p)	Contrastive	FiLM (demo), Concat (lang)	DeL-TaCo (ours)	25.2 ± 2.9
–	DistilBERT (p)	–	FiLM	Language-only	9.4 ± 1.8
CNN	DistilBERT (p)	Contrastive	FiLM (demo), Concat (lang)	DeL-TaCo (ours)	$\mathbf{27.1 \pm 2.5}$
–	CLIP (p) + MLP head	Contrastive	FiLM	DeL-TaCo (ours)	10.7 ± 2.8
CLIP (p)	CLIP (p) + MLP head	Contrastive	FiLM (lang), Concat (demo)	DeL-TaCo (ours)	11.8 ± 3.2
Demo Encoder Comparisons					
R3M (p) + MLP head	miniLM (p)	Contrastive	FiLM (demo), Concat (lang)	DeL-TaCo (ours)	13.5 ± 3.3
Task Encoder Loss Ablations					
CNN	miniLM (p)	Cosine distance	FiLM (demo), Concat (lang)	DeL-TaCo (ours)	12.5 ± 2.2
CNN	miniLM (p)	None	FiLM (demo), Concat (lang)	DeL-TaCo (ours)	10.0 ± 3.3
Task Conditioning Architecture Ablations					
CNN	miniLM (p)	Contrastive	Concat (demo + lang)	DeL-TaCo (ours)	7.9 ± 4.5
CNN	miniLM (p)	Contrastive	FiLM (demo + lang)	DeL-TaCo (ours)	22.4 ± 2.3

miniLM as our language encoder. We compare with other language models such as DistilRoBERTa (Sanh et al., 2020) and DistilBERT (Sanh et al., 2019). These were chosen because they have (1) a relatively small number of parameters for computational efficiency, and (2) were shown to achieve high performance on language-conditioned robotic policies when compared to other common language models (Mees et al., 2022). When using CLIP as the demo and language encoder, we did not notice much of a performance difference from adding a finetuneable MLP head on top of the frozen CLIP language encoder. We ultimately decided to use miniLM for all of our experiments in Tables 3.6 to 3.9 because it performed better than DistilBERT over the average of the language-only and DeL-TaCo task conditioning methods, and more crucially, was empirically more stable during demo encoder training than DistilBERT.

Demonstration Encoder Model Comparison. Using pretrained demonstration encoders, such as R3M, did not perform better than training a small CNN from scratch—a similar takeaway from our experiments with pretrained CLIP.

Loss Ablations. Our approach outperforms using a BC-Z-styled cosine-distance loss term $\mathcal{L}_{demo}(z_{demo}, z_{lang}) = 1 - z_{demo} \cdot z_{lang}$ (where $\|z_{demo}\| = \|z_{lang}\| = 1$). Our contrastive task encoder loss term is crucial; without it (setting $\alpha_d = 0$ in Eqn. 3.3), performance drops substantially.

Task Conditioning Architecture Ablations. Concatenating z_{demo} and z_{lang} before feeding into FiLM layers causes a slight drop in performance compared to our architecture in Figure 3.2, and simply concatenating z_{demo} and z_{lang} to the image observation embeddings without FiLM dramatically decreases performance.

3.7 Conclusion

When specifying tasks through language or demonstrations, ambiguities can arise that hinder robot learning, especially when the demonstrations or instructions were provided in an environment that does not perfectly align with the environment the robot is in. In this paper, we create a benchmark of learning 300 highly diverse pick-and-place tasks and propose a simple framework, DeL-TaCo, to resolve ambiguity during task specification by using both language and demonstrations during both training and testing.

Two main obstacles to deploying household robotic systems are the inability to generalize to new environments and tasks, and prohibitively high end-user effort needed to teach robots these new tasks. Our results show progress on both fronts: over previous task-conditioning methods, DeL-TaCo improves generalization performance over previous SOTA unimodal-conditioned approaches by roughly $5-18\%$ and reduces human effort on our benchmark by roughly 50 expert demonstrations per task.

3.8 Limitations and Future Work

Our work leaves a number of areas for improvement. First, we experiment only with pick-and-place tasks. Future work may need hierarchical policies and encoders to learn more diverse manipulation skills and temporally-extended tasks. Second, we experimented with a rigid set of template-based language instructions for each task, but we leave experimenting with our collected dataset of diverse, human-provided instructions to future work. Third, DeL-TaCo may not find sufficient value in language for tasks that require high-precision spatial movements, or if the visual

demonstration modality already perfectly disambiguates the task. Fourth, we did not find pretrained vision-language models, such as CLIP, to increase performance in our simulation-based environment, most likely because of the domain mismatch between our simulation objects and the more real-world-centric images CLIP was trained on. Investigating ways to better leverage pretrained vision-language models for multimodal task specification, in tandem with real-world robotic tasks and real-world human demonstrations, would be a promising line of future research.

Chapter 4: Natural Language Can Help Bridge the Sim2Real Gap

In Chapter 3, we saw how a simple language instruction could contain information equivalent to 50 demonstrations when training robots to perform new tasks. We also saw that more broadly, teaching robots new tasks is often better done with both demonstrations and language, rather than a single modality alone. In this chapter, we further extend the idea of using language as a store of meaning to learn from scarce data, but instead of few-shot generalization to new tasks, we demonstrate how language can enable few-shot generalization to new domains. This chapter represents work published in RSS 2024 (Yu et al., 2024).

4.1 Introduction

Recently, visual imitation learning (IL) has achieved significant success on manipulation tasks in household environments (Schaal, 1999; Brohan et al., 2022). However, these methods rely on large amounts of data in very similar domains to train data-hungry image-conditioned policies (Brohan et al., 2022, 2023; Padalkar et al., 2023). Some researchers are attempting to generalize visual IL to any target domain by collecting large datasets of demonstrations from mixed domains. In this work, we explore a different approach: can we transfer a policy trained on cheaply acquired, diverse simulation data to a real-world target task with just a few demonstrations?

A solution to effectively leverage cheap sim data while successfully fitting scarce real-world demonstrations is to create a domain-agnostic visual representation and use it for policy training. Such a representation should enable the policy to use the simulation image-action data as an inductive bias to learn with few-shot real world data. This representation must allow the policy to tap into the right distribution of actions by being broad enough to capture the task-relevant semantic state

from image observations, yet fine-grained enough to be conducive to low-level control. For instance, a sim and real image observation, both showing the robot gripper a few inches above a pan handle, should lie close together in the image embedding space to lead to similar actions, even if the two images have large differences in pixel space.

How might we acquire supervision for learning such a visual representation? Language is an ideal medium for providing it. Descriptions of task-relevant features in image observations, such as whether or not a gripper is close to a pan handle, serve as a unifying signal to align the representations of images between sim and real. We hypothesize that if a sim and real image have similar language descriptions (e.g., "the gripper is open and right above the pan handle"), then their underlying semantic states are also similar, and thus the actions the policy predicts conditioned on each image should also be semantically similar (e.g., moving downward to reach the pan handle). The pretrained embedding space of large language models (LLMs) offers a well-tuned signal that can be leveraged to measure the semantic similarity between real and sim images via their associated language descriptions (see

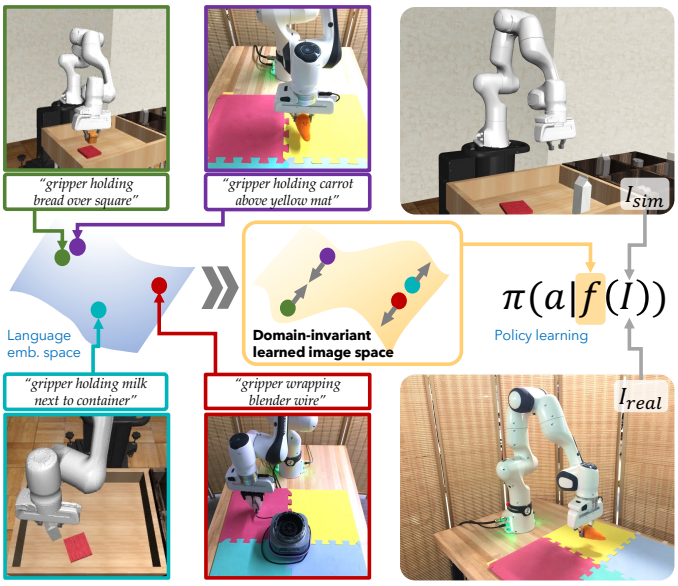


Figure 4.1: **Bridging the sim2real gap with language**. Robot images from simulation and the real world with similar language descriptions (*green & purple borders*) are mapped to similar features in language embedding space, while sim and real images with different language descriptions (*teal & red*) are mapped to faraway locations. We propose using language embedding similarities to re-shape the image embeddings (*center*) to create a domain-invariant image space. A policy is learned conditioned on these image embeddings from both sim and real images (*right*).

Fig. 4.1). This simple insight allows us to learn a domain-agnostic visual representation to bridge the visual sim2real gap.

A popular paradigm in foundation model research is to first pretrain the backbone on large datasets, and then add and train a task-specific head to process the backbone outputs to perform a downstream task. We borrow from this paradigm by first pretraining an image encoder to predict the pretrained embeddings of language descriptions of images from roughly a few hundred trajectories in sim and real, with language labels on each image. Then we use this image encoder as the backbone of our IL policy and train on action-labeled data from both the sim and real domains simultaneously, where we only need a few action-labeled demonstrations from the real world.

In this paper, we introduce Lang4Sim2Real, a lightweight framework for transferring between any two domains that have large visual differences but contain data across a similar distribution of tasks. Our approach has the following main advantages over prior sim2real efforts:

1. Alleviates the need for engineering-intensive system identification, or more broadly needing to exactly match a sim environment to the real environment both visually and semantically.

2. Enables sim2real transfer on tasks involving deformable objects that are hard to simulate with the same dynamics and visual appearance as the real-world version of the objects.

3. Bridges a wide sim2real gap that includes differences in: camera point-of-view (1st vs 3rd person), friction and damping coefficients, task goals, robot control frequencies, and initial robot and object position distributions.

In the few-shot setting, on long-horizon multi-step real-world tasks, these advantages enable Lang4Sim2Real to outperform prior SOTA methods in sim2real and

vision representation learning by 25-40%. To our knowledge, this is the first work that shows that using language to learn a domain-invariant visual representation can help improve the sample efficiency and performance of sim2real transfer.

4.2 Related Work

Our main contribution is a method to learn domain-invariant image representations by exploiting natural language descriptions as a bridge between domains for sim2real transfer. While we believe this has not been explored before, significant related research has been done in vision-language pretraining, sim2real techniques, and domain-invariant representations for control.

4.2.1 Vision Pretraining for Robotics

Various works have found that **vision-only pretraining** improves performance on image-based robotic policies. Prior work has explored pretraining objectives ranging from masked image modeling (Radosavovic et al., 2023), image reconstruction (Zhao et al., 2022; Gupta et al., 2022b; Seo et al., 2023), contrastive learning (Laskin et al., 2020; He et al., 2020), video frame temporal ordering (Jing et al., 2023), future frame prediction (Zhao et al., 2022), and image classification (Yuan et al., 2022; Wang et al., 2022a) on internet-scale datasets such as ImageNet (Deng et al., 2009), Ego4D (Grauman et al., 2021), Something-Something (Goyal et al., 2017), and Epic Kitchens (Damen et al., 2018). While these vision-only pretraining objectives learn good representations for robotic control within a specific domain distribution (such as the real world), they are not necessarily robust to the wide domain shifts encountered during sim2real.

In **vision-language pretraining**, contrastive learning (Radford et al., 2021; Zhai et al., 2021) has been shown to learn valuable representations for robotic tasks (Shridhar et al., 2021, 2022). However, these pretrained visual representations are often overly influenced by the semantics of language captions. This results in a represen-

tation that is too object-centric to differentiate between different frames of a robot demonstration, lacking the level of granularity needed for spatial-temporal understanding. R3M (Nair et al., 2022) addresses this by learning semantics from language labels of videos but also training with a time contrastive loss between video frames. Prior work in multimodal representations (Zhu et al., 2023) found language to be effective in aligning representations learned across multiple modalities including depth and audio. Instead of using language to bridge modalities, our approach uses language to bridge visual representations between domains.

4.2.2 Sim2Real

While we approach sim2real through vision-language pretraining, there are many alternative, well-researched techniques. **Domain randomization** (Andrychowicz et al., 2020; Matas et al., 2018; Tobin et al., 2017) involves varying physical parameters and visual appearances of the simulation to train a policy that functions in a wide distribution of domains that hopefully also covers the target domain distribution. However, domain randomization requires a large amount of diverse training data and attempts to be simultaneously performant in an overly broad distribution of states, leading to a suboptimal and conservative policy that takes longer to train. **System identification** (Yu et al., 2017; Kaspar et al., 2020) involves tuning the simulation parameters to match the real world in order to create a custom-tailored simulation environment that easily transfers to the real domain. However, this process is very engineering intensive and time consuming, and it may be intractable to simulate all real world physical interactions with high fidelity and throughput. In contrast, our sim2real approach can handle large source and target domain discrepancies with a few target task demonstrations and does not require system identification or domain randomization.

4.2.3 Domain-Invariant Representations

Several methods have been proposed to learn domain invariant representations. The domain-adaptation community has extensively researched using **Generative Adversarial Networks (GANs)** to map images from one distribution into another, using pixel space as a medium of common representation (James et al., 2019; Bousmalis et al., 2017; Ho et al., 2021; Rao et al., 2020). However, GANs require a large training dataset and are notorious for unstable training. Additionally, enforcing similarity on the input image side at the pixel level is less efficient than our method, which encourages cross-domain distributional similarity in a compact, low-dimensional image encoder space. Furthermore, researchers in self-driving have studied using **semantic segmentation** and depth maps (Müller et al., 2018; Ai et al., 2023) as a common representation space between domains, though their effectiveness has only been demonstrated in navigation tasks with binary segmentation masks, which is too simplified for the long-horizon manipulation tasks we consider.

4.2.4 Language and Robotics

A growing body of work has investigated training **multitask robotic policies** conditioned on language instruction embeddings (Jang et al., 2021; Lynch and Sermanet, 2021; Mees et al., 2021, 2022; Shao et al., 2020; Sodhani et al., 2021; Silva et al., 2021; Karamcheti et al., 2021), or a combination of language instructions and goal images/demonstrations (Jiang et al., 2022; Shah et al., 2023; Yu and Mooney, 2022). Our approach also involves learning a language-conditioned policy, but unlike prior work, our main novelty is using language for a second use-case: as scene descriptors during pretraining to pull together semantically similar image observations between two visually dissimilar domains. Language has also been used for **reward shaping** in RL (Nair et al., 2021b; Goyal et al., 2019, 2020; Fan et al., 2022; Ma et al., 2023), and as a high-level planner in long-horizon tasks (Huang et al., 2022; Ahn et al., 2022; Chen et al., 2022; Raman et al., 2023). These areas of research

are more ancillary to our contributions, as we demonstrate our approach with IL instead of RL and with fine-grained manipulation tasks that do not require extensive planning.

4.3 Problem Description

In this work, we address the problem of few-shot visual imitation-learning (IL): learning a visuomotor manipulation policy in the real world based on a few real-world demonstrations. We assume access to a large amount of simulation data and cast few-shot IL as a sim2real problem. More concretely, we render the few-shot IL problem as a $k+1$ multi-task IL problem: k tasks from simulation and the target task (with a few demonstrations) in the real world. In general terms, we assume a *source domain* in which data can be acquired cheaply and a *target domain* where data is expensive to collect.

In our setting, we consider access to two datasets across two domains: $\mathcal{D}^{\text{s}}$, which spans multiple tasks in the source domain, and $\mathcal{D}^{\text{t}}_{\text{target}}$, which contains a small number of demonstrations of the target task in the target domain we want to transfer to. Thus, we assume that $|\,\mathcal{D}^{\text{s}}\,| >> |\,\mathcal{D}^{\text{t}}_{\text{target}}\,|$, due to how expensive target domain data collection is (such as in the real world). We make two assumptions about the two domains. First, we assume the source and target tasks are all of the same general structure, such as multi-step pick-and-place task compositions, but with different objects and containers across different subtasks. Otherwise, transfer would be infeasible in the low-data regime if the source and target domain tasks lack similarity. Second, to train a common policy for both domains, we assume the domains share state and action space dimensionality. We make no further assumptions about the similarity between the two domains.

All of our datasets are in the form of expert trajectories. Each trajectory, $\tau = \{(I_t, s_t, [a_t, l_t], l_{task})\}$, is a sequence of tuples containing an image observation, I_t (128×128 RGB), robot proprioceptive state, s_t (end effector position and joint

angles), and a language instruction of the task, l_{task}. Note that l_{task} is the same over all timesteps of all trajectories in a given task. $[a_t, l_t]$ denotes that a trajectory may optionally also include robot actions (in which case we consider the trajectory a full demonstration) and/or a language description of the image I_t. In the following sections, we identify with $\tau[L]$ a trajectory with language descriptions l_t, but no actions a_t. Similarly, $\tau[A]$ is a full demonstration with actions, a_t, but no language descriptions, l_t.

The language labels for images can be automatically generated from a programmatic function that maps image observations to language scene descriptions depending on the relative position between the robot and the objects in the scene. We elaborate on these language labels and how to automatically collect them in Section 4.4.1. Note that these language *scene descriptions*, l_t, are different from the language *instruction* associated with each task, l_{task}.

Different data elements and types of trajectories will be used during pretraining and policy few-shot training: during pretraining, we use $\tau[L]$ image-language (I_t, l_t) pairs from $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$. During policy learning, we use $\tau[A]$ data: $(I_t, s_t, a_t, l_{task})$ tuples from $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$. In the next section, we explain how these two steps are defined for Lang4Sim2Real.

4.4 Lang4Sim2Real: Few-Shot IL with Sim&Real

In our method, we adopt the common *pretrain-then-finetune* learning paradigm (see Fig. 4.2). First, we pretrain an image backbone encoder on cross-domain language-annotated image data (Sec. 4.4.2). Then, we freeze this encoder and train a policy network composed of trainable adapter modules and a policy head to perform behavioral cloning (BC) (Schaal, 1999) on action-labeled data from both domains (Sec. 4.4.3). To leverage the simulation data, we train a $k+1$ multi-task BC policy that learns for k tasks in the source domain (sim) and 1 in the target domain (real, few shot).

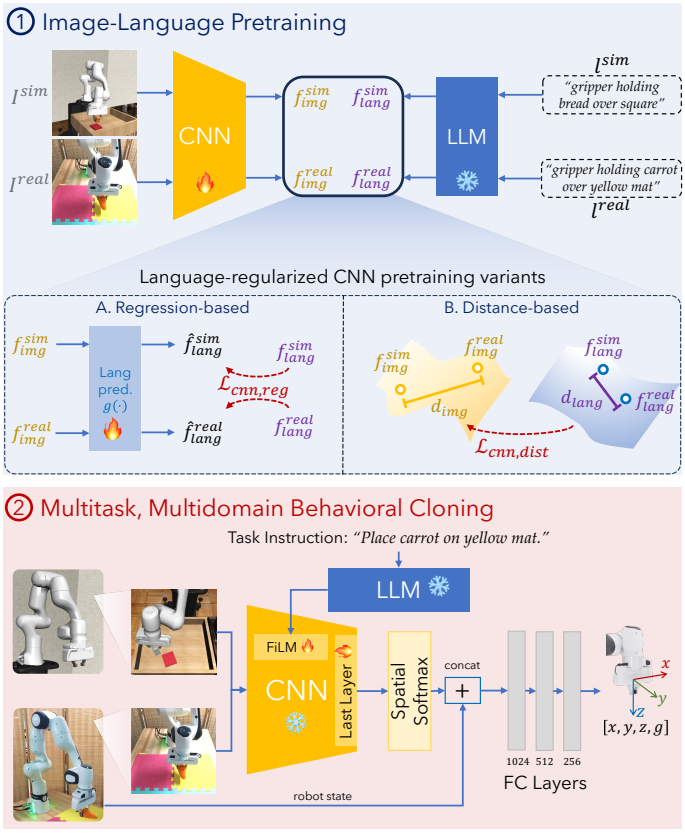


Figure 4.2: Method. (i) Top: During Image-Language Pretraining, we train the image encoder f_{cnn} using the language embeddings associated with descriptions of both sim and real image observations. f^d_{img} and f^d_{lang} refer to the output features of the CNN and the LLM, respectively, in domain d. With regression-based loss (A) the image embeddings are pushed to predict the corresponding language embeddings whereas with distance based loss (B) the pair of image embeddings is pushed together/apart based on the similarity of the language embeddings. **(ii) Bottom**: During Multitask, Multidomain BC, we freeze our pretrained f_{cnn}, add adapter modules and a policy head and allow the last layer of the CNN to finetune, then train the resulting multitask language-conditioned policy on $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$.

4.4.1 Automatic Language labeling of Images

To acquire image-language pairs for pretraining, we implement an automated pipeline for labeling the images of a trajectory that occurs synchronously during scripted policy demonstration collection (see Appendix B.1). Each if-case in the scripted policy corresponds to a stage index, where in pick-and-place, the first stage corresponds to the gripper moving to a point above the object, the second stage corresponds to the gripper moving vertically down toward the object, and so on. We

define a list of template strings describing the scene for each of these stages, so the stage indexes into the template string list, giving us our language annotation for the image. See Table 4.3 for all template strings, and Sec. 4.4.5.1 for details about our language labeling procedure.

However, our language labeling process need not be synchronously coupled with scripted policy demonstration collection. We also implement an alternative labeling process that leverages off-the-shelf vision-language models to detect the location of objects and the gripper in the image to predict the stage index. This process runs on previously-collected trajectories and requires only the images of a trajectory alone, without additional action or state information. We describe this second process in Sec. 4.4.5.2. Empirically, using language from this second, more scalable automated approach does not degrade the performance of our method.

4.4.2 Cross-Domain Image-Language Pretraining

After collecting trajectories with language labels, our first step in Lang4Sim2Real involves learning a domain-invariant representation that will enable leveraging simulation data for few shot IL. For that, we need to learn an image observation encoder, $f_{cnn} : I_t \rightarrow \mathbb{R}^{d_{cnn}}$, that attains the following property: it should preserve the semantic similarity of scenes in images between the two domains. For instance, if both image I^s from $\mathcal{D}^{\text{s}}$ (sim) and image I^t from $\mathcal{D}^{\text{t}}_{\text{target}}$ (real world) show the robot's gripper open and a few inches above the object to grasp, even if from different viewing angles, then we want their image embeddings to be close together in the learned image encoding space. This will facilitate policy learning later, as the policy will need to draw from a similar distribution of actions for similar scene semantics, which are now already mapped into similar visual features.

Theoretically, off-the-shelf pretrained vision-language models (VLMs) (Radford et al., 2021; Nair et al., 2022) should already possess these properties as they were trained on a massive distribution of image and language data. However, in the

context of robot manipulation, pretrained VLMs tend to encode all observations of the trajectory into a very narrow region of the embedding space without sufficient distinction for task-relevant, semantic aspects of the image such as the location of the gripper in relation to the manipulated objects. This renders them unsuitable without additional finetuning for our application (see Sec. 4.6).

In Lang4Sim2Real, we propose an alternative approach to obtain a visual representation with the aforementioned desired property. We train a ResNet-18 (He et al., 2015) from scratch as our image encoder using image-language tuples (I^s, l^s) from $\mathcal{D}^{\text{s}}$ and (I^t, l^t) from $\mathcal{D}^{\text{t}}_{\text{target}}$. We denote this vision language pretraining dataset as $\mathcal{D}_{VL} = \{(I^d, l^d) : (I^d, l^d) \in \mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}\}$, where d is either the source or target domain. The images are observations collected during 100 demonstrations from each of the tasks in $\mathcal{D}^{\text{s}}$ and 25-100 demonstrations from $\mathcal{D}^{\text{t}}_{\text{target}}$, totaling around 10k images per domain. We assume that the two sets of language descriptions in $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t}}_{\text{prior}}$ are similarly distributed; otherwise, language may not help learn domain-invariant features between $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t}}$.

To effectively leverage language as a bridge between visually different domains, we need a well-tuned (frozen) language model, $f_{lang} : l \rightarrow \mathbb{R}^{d_{lang}}$, to map strings to $d_{lang}-$dimensional language embeddings. We use off-the-shelf miniLM (Wang et al., 2020), since prior work (Mees et al., 2022) has demonstrated its effectiveness for language-conditioned control policies compared to other small, off-the-shelf language models.

Given the data and the language embedding described above, we propose two variants in Lang4Sim2Real for the image-language pretraining step that can obtain a sim-real agnostic representation based on language supervision (see Fig. 4.2(i)A-B).

4.4.2.1 Variant A: Language-Regression

Our first variant is a straightforward use of language supervision to shape the image embedding space: predicting the language embedding of the description, l^d,

given the embedding of the corresponding image, I^d. We sample image-language pairs from the $\mathcal{D}_{VL}$ dataset defined above: $(I^d, l^d) \sim \mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$. Let $g : \mathbb{R}^{d_{cnn}} \rightarrow \mathbb{R}^{d_{lang}}$ be a single linear layer (language predictor in Fig. 4.2(i)(A)) trained to minimize the following loss:

$$\mathcal{L}_{cnn,reg}(\mathcal{D}_{VL}) = \left\| g\left(f_{cnn}(I^d)\right) - f_{lang}(l^d) \right\|_2^2 \tag{4.1}$$

We use the loss to train both the language predictor and the CNN backbone. The loss provides strong language supervision by encouraging f_{cnn} to directly regress toward the frozen language embeddings of the image descriptions, effectively making the pretrained image encoder reflect the LLM embedding space.

4.4.2.2 Variant B: Language-Distance Learning

We also experiment with a second variant of image-language pretraining that incorporates language with a softer form of supervision. We posit that the exact values of the language embeddings do not themselves convey meaning; rather, the semantic similarity between two images lies in the pairwise distances between their corresponding two language embeddings. Thus, we design an objective to regress the image embedding distances between a pair of images from the two domains to their corresponding language distance:

$$\mathcal{L}_{cnn,dist}(\mathcal{D}_{VL}) = \left\| f_{cnn}^{\top}(I^s) f_{cnn}(I^t) - d\big(l^s, l^t\big) \right\|_2^2 \tag{4.2}$$

where the language distance function we use, $d : l \times l \rightarrow \mathbb{R}$ is BLEURT (Sellam et al., 2020), a learned similarity score between two strings commonly used in the NLP community. We normalize $d(\cdot, \cdot)$ between 0 and 1 for all possible (l^s, l^t) pairs in our image-language dataset, where 1 indicates the highest similarity between any two strings in the dataset. Empirically, over our set of language descriptions, we found BLEURT provided a richer signal than simply taking dot products or ℓ-2 distances between language embeddings. The output of f_{cnn} is unit normalized before taking the dot product. We compare both variants (see Sec. 4.6) to assess whether the

additional degrees of freedom from the looser distance supervision are beneficial later on for policy training.

4.4.3 Multitask, Multidomain Behavioral Cloning

Our second step in Lang4Sim2Real involves learning a multi-domain, multi-task, language-conditioned BC policy (see Fig. 4.2(ii)). By leveraging our learned domain-invariant representation for robotic control, this policy should be able to perform well in the real-world target task with only a few demonstrations, thanks to the additional information it can extract from simulation.

During this phase of policy learning, we freeze all but the last layer to preserve the semantic scene information encoded in the learned, domain-invariant representation, f_{cnn}, while enabling the network to adapt to the new downstream task of low-level control. We also insert trainable FiLM layer blocks (Perez et al., 2018) as adapter modules in f_{cnn} to process the language instruction embeddings between the frozen convolution layers. Finally, we include a few fully-connected layers as a policy head to process the image feature, $f_{cnn}(I_t)$, and proprioceptive state, s_t, and train the resulting policy π with BC loss to predict the mean and standard deviation of a multivariate Gaussian action distribution, as described below.

Let our training dataset $\mathcal{D}_{BC} = \mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$ be a set of demonstrations τ^d, for domain $d \in \{\text{source, target}\}$. As explained in Sec. 4.3, each demonstration is a sequence of tuples $x_t = \left(I_t^d, s_t^d, a_t^d, l_{task}\right)$ containing the image observation, proprioceptive state, language instruction for the task, and action at timestep t. We train with the following standard BC negative log probability loss (Pomerleau, 1988):

$$\mathcal{L}_\pi(\mathcal{D}_{BC}) = \frac{1}{B} \sum_{\substack{x_t \sim \tau^d \\ \tau^d \sim \mathcal{D}_{BC}}} -\log \pi \left(a_t^d \middle| f_{cnn}(I_t^d), s_t^d, l_{task}\right) \tag{4.3}$$

where B denotes the batch size.

The policy is trained on $k+1$ tasks: k from $\mathcal{D}^{\text{s}}$ (thousands of trajectories per task) and 1 from $\mathcal{D}^{\text{t}}_{\text{target}}$ (≤ 100 trajectories, see Sec. 4.5). In each batch, we sample

m tasks uniformly at random from the $k+1$ tasks, and then query $\mathcal{D}_{BC}$ for a fixed number of transitions from trajectories for each of the m selected tasks.

We hypothesize that cross-domain image-language pretraining (Sec. 4.4.2) improves policy learning because it helps ensure that image observations of different domains depicting semantically similar scenes map into similar regions of the learned embedding space. This accelerates learning not only on $\mathcal{D}^{\text{s}}$ data but also helps alleviate data scarcity in $\mathcal{D}^{\text{t}}_{\text{target}}$ because the pretrained image backbone encodes $\mathcal{D}^{\text{t}}_{\text{target}}$ images into an in-distribution region of the learned image embedding space, alleviating common issues with visual distribution shift and enabling our method to leverage simulation data to compensate for the lack of real-world action-labeled data, improving sim2real transfer.

4.4.4 Detailed Policy Network Architecture & Hyperparameters

For the policy backbone, we use a ResNet-18 architecture but made changes to the strides and number of channels to adapt the network to our $128 \times 128 \times 3$ image size. Hyperparameters are shown in Table 4.1. A detailed layer-by-layer architecture figure of our policy is shown in Figure 4.3. During policy training, only the last CNN layer, FiLM blocks, and policy head (FC layers) are finetuned, while all other layers are kept frozen.

Table 4.2 shows our BC training hyperparameters. In each training iteration, we sample 4 random tasks from our training buffer and get 57 samples per task, for a total batch size of 228.

Table 4.2: **Imitation learning hyperparameters.**

Attribute	Value
Number of Tasks per Batch	4
Batch Size per Task	57
Learning Rate	3×10^{-4}

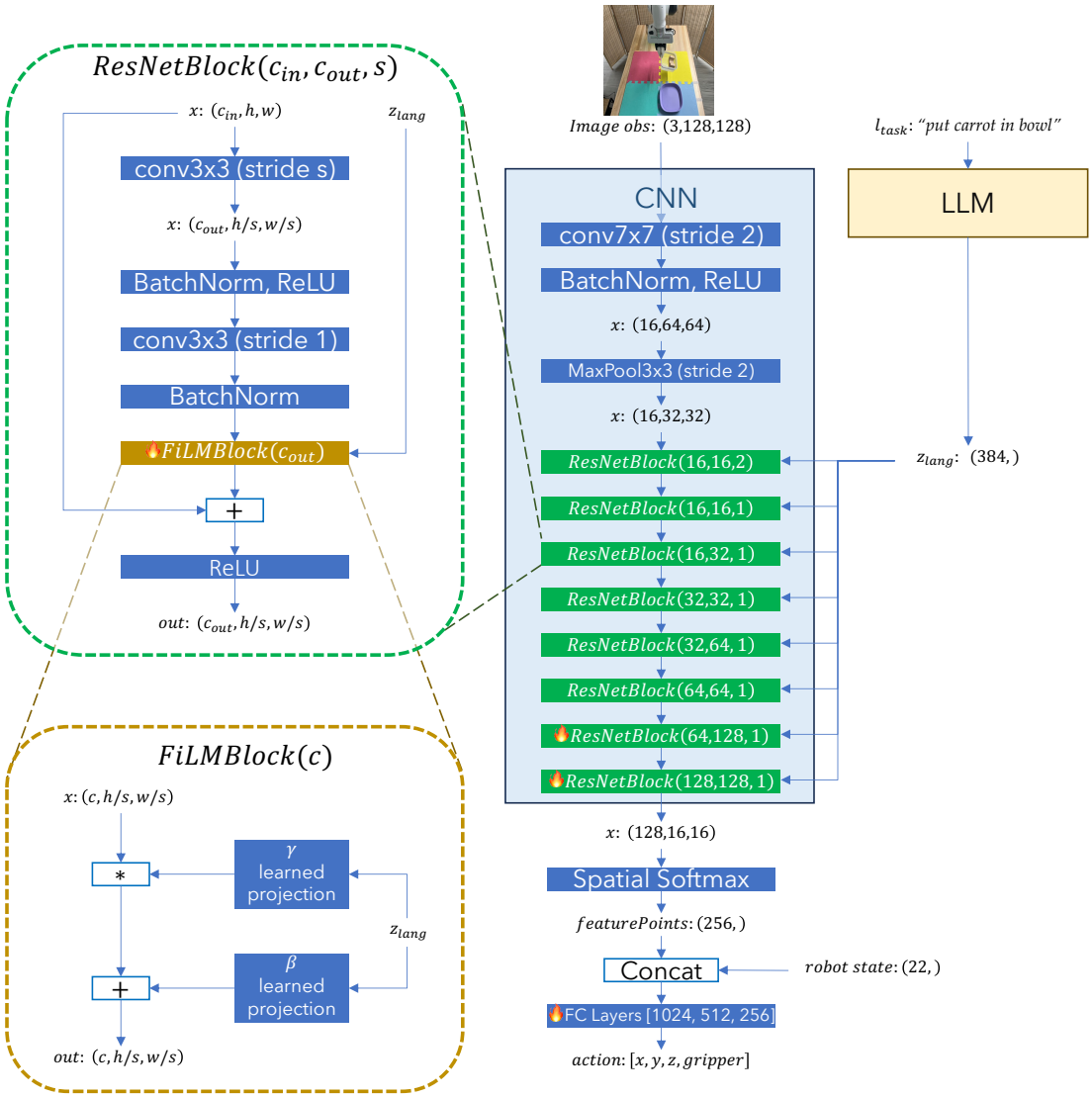


Figure 4.3: Detailed Policy Network Architecture. Fire denotes layers trained during policy learning. The early CNN modules are kept frozen to maintain the intermediate representations learned from the pretraining phase.

Table 4.1: **Policy π hyperparameters.**

Attribute	Value
Input Height	128
Input Width	128
Input Channels	3
Number of Kernels	[16, 32, 64, 128]
Kernel Sizes	[7, 3, 3, 3, 3]
Conv Strides	[2, 2, 1, 1, 1]
Maxpool Stride	2
Fully Connected Layers	[1024, 512, 256]
Hidden Activations	ReLU
FiLM input size	384
FiLM hidden layers	0
Spatial Softmax Temperature	1.0
Learning Rate	3×10^{-4}
Policy Action Distribution	Multivariate Isotropic Gaussian $\mathcal{N}(\mu, \sigma)$
Policy Outputs	(μ, σ)
Image Augmentation	Random Crops
Image Augmentation Padding	4

4.4.5 Language Labeling of Image Observations Details

4.4.5.1 Language Labeling During Scripted Policy

We automatically label image observations with language descriptions during the scripted policy data collection process. Each image is assigned a stage number based on the if-case of the scripted policy, which corresponds to a semantic positional arrangement between the gripper and the relevant objects on the scene. Stage numbers map 1-to-1 to the template language strings shown in Table 4.3.

For example, for the pick-and-place/stack object task, we define 7 stages and 7 corresponding language string templates, where the first stage is when the gripper moves toward a point above the object, the second stage is when the gripper moves downward toward the object, and so on. For the 2-step pick-and-place task, we use 14 stages—2 consecutive lists of the 7 individual pick place string templates, where the

object and container variables of each template are filled in with the proper names.

4.4.5.2 Alternative Approach: Language labeling with off-the-shelf VLMs

To relax the requirement that our automated language labeling process must occur synchronously with a scripted policy collecting demonstrations, we implemented an alternative approach that is decoupled from the demonstration collection process. First, we use an off-the-shelf open-vocabulary object detector model, GroundingDINO (Liu et al., 2023b), to output bounding boxes for the relevant objects on the scene. No finetuning of GroundingDINO is required. Second, we train a CNN-based gripper state predictor to predict the gripper position (x, y, z) as well as whether the gripper is opened or closed in a given image. This network is trained on previously collected (image, gripper position, gripper opened/closed) data from 100 trajectories, and takes one minute to train on a single A5000 GPU. Using these two models, we can get the gripper state and position relative to the objects, enabling us to predict a stage number that corresponds fairly closely with the actual stage number as outputted by our scripted policy. Finally, we verified that training our method on VLM-derived language annotations does not degrade performance. We performed image-language pretraining with language labels from either labeling method and tested on 2-step pick-and-place with 100 real-world trajectories. Both methods achieve 90% success rate averaged over 2 seeds.

4.5 Experimental Setup

We evaluate Lang4Sim2Real in two settings: a `sim2sim` setting where we test the transfer abilities between two simulated domains with visual and physical differences, and the `sim2real` setting, where the few shot IL is defined in the real world and we use simulation to address the data scarcity. `Sim2sim` serves as a platform to evaluate in depth the behavior of Lang4Sim2Real with a fully controlled domain gap, while `sim2real` is our setting of interest for this work. We will use three task

suites that we explain below. See Figure 4.5 for detailed frame rollouts of each task. In a slight overload of notation from Sec. 4.3, here we use $\mathcal{D}^{s}$ and $\mathcal{D}^{t}$ to denote the source and target domains, respectively.

4.5.1 `Sim2Sim` and `Sim2Real` Environment Differences

In `sim2sim`, $\mathcal{D}^{s}$ and $\mathcal{D}^{t}$ are both sim environments with large differences in camera point-of-view (third person vs. first person), joint friction, and damping:

1. **Camera point-of-view**: $\mathcal{D}^{s}$ image observations are third person (looking toward the robot), and $\mathcal{D}^{t}$ image observations are first person (over the shoulder), a large change of viewing angle.
2. **Friction and Damping**: Joint friction and damping coefficients are $5\times$ and $50\times$ higher in $\mathcal{D}^{t}$ than $\mathcal{D}^{s}$, which significantly changes the dynamics.

In the `sim2real` setting, we employ a setup with a wide sim2real gap that we aim to bridge using language that includes differences in control frequency, task goals, visual observation appearance, objects, and initial positions:

1. **Control frequency**: The simulated $\mathcal{D}^{s}$ policy runs at 50Hz while the real world $\mathcal{D}^{t}$ policy runs at 2Hz.
2. **Objects**: The objects on the scene in each task differ between simulation and real data, except the robot itself.
3. **Visual Observation**: Backgrounds and camera angles are markedly different between the two domains.
4. **Initial positions**: The initial object and robot positions are different across sim and real.

4.5.2 Evaluation Metrics

For all `sim2sim` and `sim2real` experiments, we measure task success rate. In `sim2real`, this is calculated through ten evaluation trials for each of two seeds per task, for a total of 20 trials per table entry. In each set of ten trials, we place the object in the same ten initial positions and orientations, evenly distributed through the range of valid initial object positions. In `sim2sim`, we also run two seeds per setting and take a success rate averaged over 720 trials between the two seeds in the final few hundred epochs of training.

4.5.3 Data

4.5.3.1 Environments

For each of our tasks, we design simulation environments on top of Robosuite (Zhu et al., 2020) in Mujoco (Todorov et al., 2012). For the real environment, we use Operational Space Control (Khatib, 1987) to control the position of the end-effector of the robot in Cartesian space. In both simulation and real, we work with a 7-DOF Franka Emika Panda arm and use a common action space consisting of the continuous xyz delta displacement and a continuous gripper closure dimension (normalized from $[-1, 0]$). The robot proprioception space is 22-dimensional, consisting of the robot's xyz end-effector position, gripper state, and sine and cosine transformations of the 7 joint angles. The image observation space is 128×128 RGB images.

4.5.3.2 Overview of Tasks

For each task suite, we collect data from simulated domain $\mathcal{D}^{\text{s}}$ and real target domain $\mathcal{D}^{\text{t}}$ (for `sim2real`) or sim target domain $\mathcal{D}^{\text{t}}$ (for `sim2sim`). All demonstrations in sim and real are collected with a scripted policy (see Appendix B.1 for further details). Sim trajectories range from 200-320 timesteps long, operated at 50 Hz, while real trajectories run at 2 Hz and range from 18-45 timesteps. Our three task suites

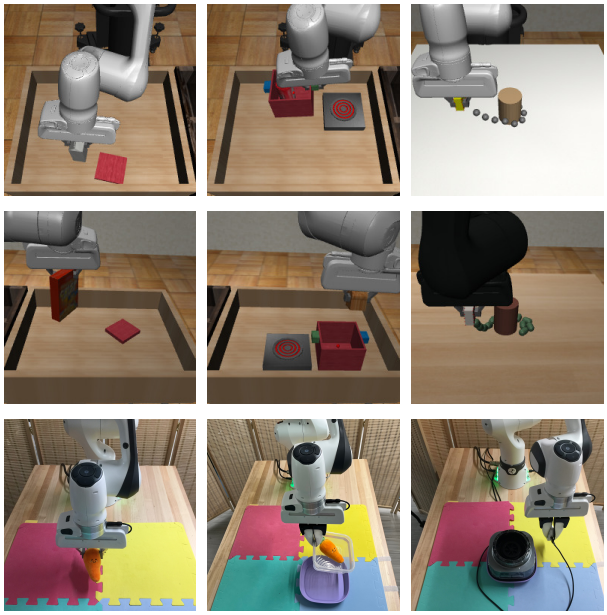


Figure 4.4: The columns depict the three task suites while each row represents an image domain. **Rows from Top to Bottom:** Simulation $\mathcal{D}^{s}$, `sim2sim` $\mathcal{D}^{t}_{\text{target}}$, `sim2real` $\mathcal{D}^{t}_{\text{target}}$. **Columns from Left to Right:** Stack Object, Multi-step Pick and Place, and Wrap Wire tasks. While similar enough to transfer prior knowledge between them, our $\mathcal{D}^{s}$ and $\mathcal{D}^{t}$ task versions have a considerable gap (Sec. 4.5.1) that we are able to bridge using language as regularization for the image representations.

allow us to test the effectiveness of Lang4Sim2Real for sim2real in a wide variety of control problems ranging from simple stacking in task suite 1, to multi-step long-horizon pick and place in task suite 2, to deformable, hard-to-simulate objects in task suite 3. See Fig. 4.4 to understand the considerable domain gap across both `sim2sim` and `sim2real` in each of the three tasks suites.

4.5.4 Task Suite 1: Stack Object

In our first suite of tasks, the robot must move an object to a target. In the simulated domain $\mathcal{D}^{s}$, the target is on top of a wooden coaster, and there are four objects: milk carton, soda can, bread, and cereal box, which correspond to the four tasks. Both the object and coaster positions are randomized over the entire workspace. We collect and train on 400 demonstrations per task (1600 total) as our $\mathcal{D}^{s}$ simulation data.

4.5.4.1 `sim2sim`

For `sim2sim` experiments on this task suite, we define a new $\mathcal{D}^{t}$ simulated environment with differences from $\mathcal{D}^{s}$ as enumerated in Sec. 4.5.1. Policies are trained with the 1600 $\mathcal{D}^{s}$ demonstrations and 100 target task $\mathcal{D}^{t}_{\text{target}}$ demonstrations.

4.5.4.2 `sim2real`

For `sim2real`, $\mathcal{D}^{s}$ remains the same as `sim2sim`. $\mathcal{D}^{t}$ is a real world environment in which the object is randomly placed on the left mat and the target task $\mathcal{D}^{t}_{\text{target}}$ is to move the object onto the right mat and open the gripper by the end of 20 timesteps.

4.5.5 Task Suite 2: Multi-step Pick and Place

Our second suite of tasks is longer-horizon. In simulation, the robot must first put an object in the pot, then grasp the pot by its handle and move it onto the stove. We categorize this as a 2-step pick-and-place task. We use the same four object-task mappings from Sec. 4.5.4. The object, pot, and stove locations are all randomized within a quadrant of the workspace. Since this task is longer horizon, we train on more data—$1,400$ trajectories per task in $\mathcal{D}^{s}$.

4.5.5.1 `sim2sim`

Similar to the stacking task, in the `sim2sim` setting, we define a new $\mathcal{D}^{t}$ environment with differences from $\mathcal{D}^{s}$ enumerated in Sec. 4.5.1 and evaluate over the four tasks when given 100 target-task $\mathcal{D}^{t}_{\text{target}}$ demonstrations.

4.5.5.2 `sim2real`

In the `sim2real` setup, $\mathcal{D}^{s}$ remains the same, while $\mathcal{D}^{t}$ is the real task of putting a carrot into a bowl, then putting the bowl onto a plate (see Fig. 4.4), and ending with the gripper open after 50 timesteps. In addition to success rate (Sec. 4.5.2), we measure average number of consecutive subtasks completed from

the beginning, allowing partial credit if the robot only succeeds in the first step of placing the carrot in the bowl. Because we only count consecutively completed subtasks, failing the first step results in a score of zero, even if the subsequent step(s) are successful.

4.5.6 Task Suite 3: Wrap Wire

Our final suite of tasks involves wrapping a long deformable wire around a fixed object. In simulation $\mathcal{D}^{s}$, we approximate a wire with a chain of spheres connected with free joints, and the task is to wrap the chain around a fixed cylinder (see Fig. 4.4). A trajectory is successful if the first link of the chain has traveled $\geq \frac{5\pi}{3}$ radians (5/6ths of a full revolution) around the cylinder. Our simulation data consists of two tasks: wrapping counterclockwise and clockwise. The initial position of the end of the chain is randomized over a region to the left of the cylinder. $\mathcal{D}^{s}$ contains 400 trajectories per task.

4.5.6.1 `sim2sim`

For our $\mathcal{D}^{t}$ sim environment, we again apply the changes specified in Sec. 4.5.1. We additionally swapped the spheres for capsules and changed the color and texture of the table, robot arm, and objects. This task has a wider sim2sim gap from additional visual and dynamics changes.

4.5.6.2 `sim2real`

In our `sim2real` experiments, the target task $\mathcal{D}^{t}_{\text{target}}$ is to first grasp the plug, then wrap the cord around the base of a blender in the middle of the workspace, and finally put the plug down, similar to what one might do before putting the appliance away. Like the sim environment, we define success if the following two conditions are met: (1) the plug travels $\geq \frac{5\pi}{3}$ radians around the blender, and (2) the plug is placed and the gripper is open at the end of 50 timesteps.

4.5.7 Baselines

To evaluate the effectiveness of Lang4Sim2Real, we consider two sets of baselines: non-pretrained baselines where the CNN is initialized from scratch, and baselines with pre-trained visual encoders. For the non-pretrained baselines, we examine training with only $\mathcal{D}^{\text{t}}$ data, and training with both $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t}}$ data. This enables us to understand the benefits of our proposed training procedure. In `sim2real`, we also compare to three popular prior sim2real approaches:

- **MMD** (Tzeng et al., 2014), which aims to minimize the distance between the mean embedding of all sim images and all real images of a batch to prevent the real images from being out-of-distribution relative to the sim images.
- **Domain randomization** (Tobin et al., 2017) of the colors, textures, and physics of the $\mathcal{D}^{\text{s}}$ environment.
- Automatic Domain Randomization with Random Network Adversary (**ADR+RNA**) (OpenAI et al., 2019), which keeps increasing/decreasing domain randomization bounds depending on the agent's performance, and also introduces a randomly initialized network for each trajectory to inject correlated noise into the agent's action conditioned on the state input.

For the pretrained baselines, we consider two strong foundation models as the visual backbone, CLIP (Radford et al., 2021) and R3M (Nair et al., 2022), commonly used visual representations for robotics that are, like our approach, also shaped by language descriptions of images/videos, and add a trainable policy head composed of fully-connected layers sharing the same dimensions as all other methods in our results.

For each task in `sim2real`, we train and evaluate with 25, 50, or 100 $\mathcal{D}^{\text{t}}_{\text{target}}$ demonstrations.

4.5.8 Our Method Variants and Ablations

In our evaluations, we compare language regression (Section 4.4.2.1) and language distance (Section 4.4.2.2), the two pretraining variants of our approach. We also ablate away the effects of language on our pretraining approach in a method called "stage classification," where the pretraining task is to predict the stage index of an image (see Section 4.4.1) instead the language embedding or embedding distance.

4.5.9 Language Description Templates of Image Observations

All template language descriptions of images for each stage and task are in Table 4.3. These are used during the pretraining phase (Sec. 4.4.2) of Lang4Sim2Real.

Table 4.3: Language Template Strings for Image Descriptions

Task	**Template String**
Pick and Place	gripper open, reaching for *{objName}*, out of *{contName}*
	gripper open, moving down over *{objName}*, out of *{contName}*
	gripper closing, with *{objName}*, out of *{contName}*
	gripper closed, moving up with *{objName}*, out of *{contName}*
	gripper closed, moving sideways with *{objName}*, out of *{contName}*
	gripper closed, with *{objName}*, above *{contName}*
	gripper open, dropped *{objName}*, in *{contName}*
Wrap Wire	gripper open, reaching for *{graspObjName}*
	gripper open, moving down over *{graspObjName}*
	gripper closing around *{graspObjName}*
	gripper closed, moving up with *{graspObjName}*
	{direction} left
	{direction} front
	{direction} right
	{direction} back
	gripper open, above *{graspObjName}* with *{flexWraparoundObjName}* fully wrapped
	gripper open, above *{graspObjName}* with *{flexWraparoundObjName}* fully unwrapped
Variable	**Possible Values**
objName	milk, bread, can, cereal, pot, carrot, bowl, bridge
contName	coaster, pot, stove, bowl, plate
flexWraparoundObjName	beads, cord, ethernet cable
graspObjName	last bead, white plug, bridge
direction	clockwise, counterclockwise

Table 4.4: `sim2real`: Performance by number of real world trajectories

Method	Action-labeled Data		Stack Object			Multi-step Pick and Place						Wrap Wire		
	Sim	Real	Success Rate (%)			Success Rate (%)			Subtasks Completed			Success Rate (%)		
	$\mathcal{D}^s$	$\mathcal{D}^t_{\text{target}}$	25	50	100	25	50	100	25	50	100	25	50	100
No Pretrain ($\mathcal{D}^t$)	–	✓	20	30	45	0	30	35	0.45	1.05	1.05	20	15	45
No Pretrain ($\mathcal{D}^s + \mathcal{D}^t$)	✓	✓	35	20	55	45	25	55	1.15	1.0	1.4	25	20	20
MMD	✓	✓	25	35	**80**	20	10	35	0.8	0.9	1.1	5	10	20
Domain Random.	✓	✓	40	60	40	10	10	25	0.7	0.6	0.7	0	0	0
ADR+RNA	✓	✓	35	30	35	15	25	40	0.85	0.8	1.3	0	10	0
Lang Reg. (ours)	✓	✓	40	**75**	**80**	**60**	**80**	**90**	**1.45**	**1.8**	**1.9**	**45**	**40**	45
Lang Dist. (ours)	✓	✓	**60**	45	**80**	55	70	75	1.35	1.65	1.6	30	25	**75**
Stage Classif.	✓	✓	40	60	60	50	60	50	**1.45**	1.55	1.5	30	**40**	50
CLIP (frozen)	✓	✓	25	5	15	10	15	40	0.3	0.45	1.0	35	35	30
R3M (frozen)	✓	✓	30	45	65	15	60	55	0.7	1.4	1.5	5	25	25

Table 4.5: `sim2sim`: Success Rate by Task (%)

Pretraining	Stack Object					Multi-step Pick and Place					Wrap Wire
	1	2	3	4	avg	1	2	3	4	avg	1
None ($\mathcal{D}^t$ data only)	15.2 ± 6.5	18.9 ± 6.7	31.9 ± 8.5	25.4 ± 9.2	22.9	20.8 ± 7.5	17.7 ± 4.4	16.3 ± 5.0	17.3 ± 8.1	18.0	69.2 ± 8.3
None ($\mathcal{D}^s + \mathcal{D}^t$ data)	22.5 ± 9.2	32.3 ± 9.8	37.9 ± 8.8	29.2 ± 8.3	30.5	28.4 ± 10.9	31.3 ± 10.7	13.9 ± 5.5	27.8 ± 10.2	25.4	82.1 ± 6.8
Lang Reg. (ours)	20.6 ± 8.1	$\mathbf{57.3 \pm 8.1}$	63.1 ± 7.7	$\mathbf{32.5 \pm 6.3}$	43.4	54.0 ± 7.2	$\mathbf{62.5 \pm 12.1}$	76.0 ± 8.7	58.5 ± 9.3	62.8	90.7 ± 5.4
Lang Dist. (ours)	23.8 ± 5.4	$\mathbf{57.3 \pm 10.6}$	66.9 ± 5.6	27.9 ± 10.8	44.0	$\mathbf{65.5 \pm 13.1}$	56.7 ± 9.9	$\mathbf{78.6 \pm 5.1}$	54.4 ± 11.5	**63.8**	90.0 ± 5.0
Stage Classif.	$\mathbf{30.4 \pm 10.4}$	52.7 ± 6.0	$\mathbf{67.5 \pm 8.3}$	27.9 ± 7.1	**44.6**	63.1 ± 9.9	62.1 ± 9.3	55.4 ± 8.5	$\mathbf{67.7 \pm 9.7}$	62.1	$\mathbf{91.4 \pm 3.6}$
CLIP (frozen)	1.7 ± 0.4	1.9 ± 1.9	3.8 ± 2.5	4.0 ± 2.7	2.9	36.1 ± 14.3	39.9 ± 8.9	28.8 ± 8.9	48.4 ± 11.9	38.3	75.6 ± 7.7
R3M (frozen)	4.5 ± 3.3	9.0 ± 4.8	19.8 ± 6.9	15.4 ± 5.4	12.2	49.4 ± 11.6	36.5 ± 11.9	47.0 ± 14.1	56.0 ± 10.0	47.2	90.2 ± 4.4

4.6 Experimental Results

Our results for `sim2sim` experiments are shown in Table 4.5, and the results for `sim2real` are shown in Table 4.4. In both tables, the methods (rows) are grouped into non-pretrained baselines, our method variants and ablations, and pretrained SOTA baselines. In `sim2real`, we additionally include a group of three rows to show the performance of prior sim2real approaches.

4.6.1 Experimental Questions and Analysis

Across the three task suites in both `sim2real` and `sim2sim`, our method generally achieves the highest success rates. To further analyze the effectiveness of our method, we pose and investigate the following experimental questions.

4.6.1.1 What is the impact of our pretraining approach?

Our method nearly doubles the success rate of both non-pretrained baselines in most task suites in `sim2real` and `sim2sim`. This indicates that Lang4Sim2Real can bridge a wide sim2real gap. One factor that may allow our method to perform well is that image observations with similar language descriptions may also have similar action labels. In Sec. 4.6.1.9, we further investigate this hypothesis with an analysis of the action distributions between images, split by their language descriptions.

Between the non-pretrained baselines, training on $\mathcal{D}^{s}$ sim demonstrations in `sim2real` provides little benefit on stack object, increases average performance by $\approx 20\%$ on multi-step pick-and-place, but decreases average performance by $\approx 10\%$ on wrap wire. However, in `sim2sim`, it provides a 10-15% increase on most tasks. This suggests that although using $\mathcal{D}^{s}$ without pretraining yields some improvement, the sim2sim gap is significant enough that pretraining is necessary for effective transfer.

4.6.1.2 How does our method compare to prior sim2real baselines?

Our method outperforms all of the prior sim2real baselines we tested against (second row-group in Table 4.4), which collectively do relatively poorly in most settings, highlighting the difficulty of the sim2real problem in our setup.

MMD averages the best performance across the three sim2real baselines and even achieves competitive performance on the easiest task of stacking an object. However, on the two other more difficult tasks, its performance does not scale well with more trajectories, which we suspect arises from stability issues in trying to push together the mean of all sim and real image embeddings in each batch. Domain randomization only exacerbates the sim2real gap since enabling all randomizations does not move the distribution of simulation trajectories closer to the real world trajectories due to the large visual dissimilarity between our simulation and real environments. ADR+RNA, which only randomizes the environment as much as possible without severely hurting the scripted policy performance, averages slightly better per-

formance than domain randomization, perhaps because the data is less diverse and easier to fit a policy to than the data from full-scale domain randomization.

4.6.1.3 How does our method compare to prior vision-language pretrained representations?

In `sim2real`, our method outperforms both pretrained baselines across the board, including R3M, which is the strongest baseline on stack object and multi-step pick-and-place. When trained on increasing amounts of real-world data, both R3M and CLIP tend to plateau—CLIP performs no better than 40% on any task, R3M has an apparent ceiling of 65%, while our method achieves up to 90%. This suggests that CLIP and R3M do not scale as well as our method when provided more data, despite being pretrained on internet-scale video and image data while our method was pretrained on images from just a few hundred sim and real trajectories.

In `sim2sim`, our method also outperforms R3M and CLIP across the board. Averaging the performance on stacking and multi-step pick-and-place, our method outperforms R3M by 15-30% and CLIP by 25-40%. On the wrap wire task, our method and R3M perform comparably, probably because the task is quite a bit easier for all methods in simulation.

4.6.1.4 What is the effect of language in learning shared representations?

We ablate the effect of language on our pretraining as the "stage classification" row in Tables 4.4 and 4.5, as mentioned in Section 4.5.8. In `sim2sim`, we see similar performance in language regression pretraining and stage classification pretraining. However, in `sim2real`, where the domain gap is larger, we see language providing a measurable benefit in all task suites, especially in multi-step pick-and-place, perhaps because pretraining with language leverages similarities in language descriptions between the first and second steps of the pick-and-place task.

4.6.1.5 How do our two image-language pretraining variants compare?

We compare our two pretraining variants introduced in Sections 4.4.2.1 and 4.4.2.2, where language regression directly predicts language embeddings while language distance is encouraged to maintain pairwise distances based on BLEURT similarity scores. Again in `sim2sim`, there is no clear winner between the two, but in `sim2real`, language regression performs better on average. This suggests that when performing language pretraining for visual representations, the more constraining regression loss outperforms the less constraining distance-matching loss on `sim2real` performance.

4.6.1.6 What is the effect of pretraining on image-language pairs where the language granularity is reduced?

To evaluate the impact of reduced language granularity on `sim2real` performance, we segment the trajectories into fewer and fewer stages, until the extreme case where the entire trajectory has only a single stage, which means that all images across all trajectories of a task have the same exact language description embedding. The language descriptions we use for each stage, for varying numbers of stages per task, are displayed in Tables B.1 (2-step pick-and-place) and B.2 (wire wrap).

Results are shown in Table 4.6. The trend is noisy, but in general, decreasing language granularity hurts performance slightly. Still, our method is robust to lower granularity, which matches our hypothesis that our pretraining approach provides significant performance gains simply by pushing sim and real images into a similar embedding distribution even if the language granularity is extremely coarse.

4.6.1.7 How does our method perform if we cannot pretrain directly on image-language pairs from the target task?

There are scenarios in which we might not have access to the real-world target task $\mathcal{D}^{\text{t}}_{\text{target}}$ during the pretraining phase, as pretraining is often done without knowledge of the downstream task. To investigate this, we introduce a real-world prior

Table 4.6: `sim2real`: Performance with Varying Language Granularity

Method	Multi-step Pick and Place						Wrap Wire		
	Success Rate (%)			Subtasks Completed			Success Rate (%)		
	25	50	100	25	50	100	25	50	100
No Pretrain ($\mathcal{D}^t$)	40	20	30	1.15	0.9	1.15	25	45	35
No Pretrain ($\mathcal{D}^s + \mathcal{D}^t$)	45	30	50	1.25	1.2	1.4	15	30	30
all-stages	**55**	**80**	**95**	**1.2**	**1.8**	**1.95**	**25**	**50**	**55**
half-stages	45	60	65	1.15	1.45	1.55	5	35	25
2-stages	35	45	75	1.05	1.3	1.6	20	**50**	40
1-stage	**55**	65	80	1.3	1.55	1.75	15	15	45
1 stage per domain	10	50	50	0.65	1.3	1.25	15	15	20

task $\mathcal{D}^t_{\text{prior}}$ that we pretrain on, and use real-world target task data $\mathcal{D}^t_{\text{target}}$ only during imitation learning. Figure 4.5 provides film strips of trajectories from the source domain data $\mathcal{D}^s$, target domain prior task data $\mathcal{D}^t_{\text{prior}}$, and target domain target task data $\mathcal{D}^t_{\text{target}}$, for each of the three task suites. The advantage of this problem setup is that we can reuse the same f_{cnn} for multiple downstream real-world target tasks as long as they are sufficiently similar to the real-world prior task. In this modified problem setup, our method still mostly outperforms all baselines, which demonstrates that our method does not overfit to the real-world task it sees during pretraining.

In Tables 4.4 and 4.5, we presented results in a setting where we both pretrained and did policy learning on two datasets, $\mathcal{D}^s$ and $\mathcal{D}^t_{\text{target}}$. In this setting, we experiment with pretraining on $\mathcal{D}^s \cup \mathcal{D}^t_{\text{prior}}$ and training our policy on $\mathcal{D}^s \cup \mathcal{D}^t_{\text{target}}$.

Our method uses extra language labels during pretraining that the baselines do not have access to. While these language labels can be acquired at scale, to compensate for this data advantage, we decided to give all baselines an augmented $\mathcal{D}^t_{\text{prior}}$ dataset that includes action-labeled demonstrations, in addition to the target task, $\mathcal{D}^t_{\text{target}}$. *Note that our method is not given* $\mathcal{D}^t_{\text{prior}}$ *action-labeled data*: it is trained only on $\mathcal{D}^t_{\text{prior}}$ images with language labels during image-language pretraining (Sec. 4.4.2) but not during BC policy learning. Therefore, the baselines in a sense serve as upper

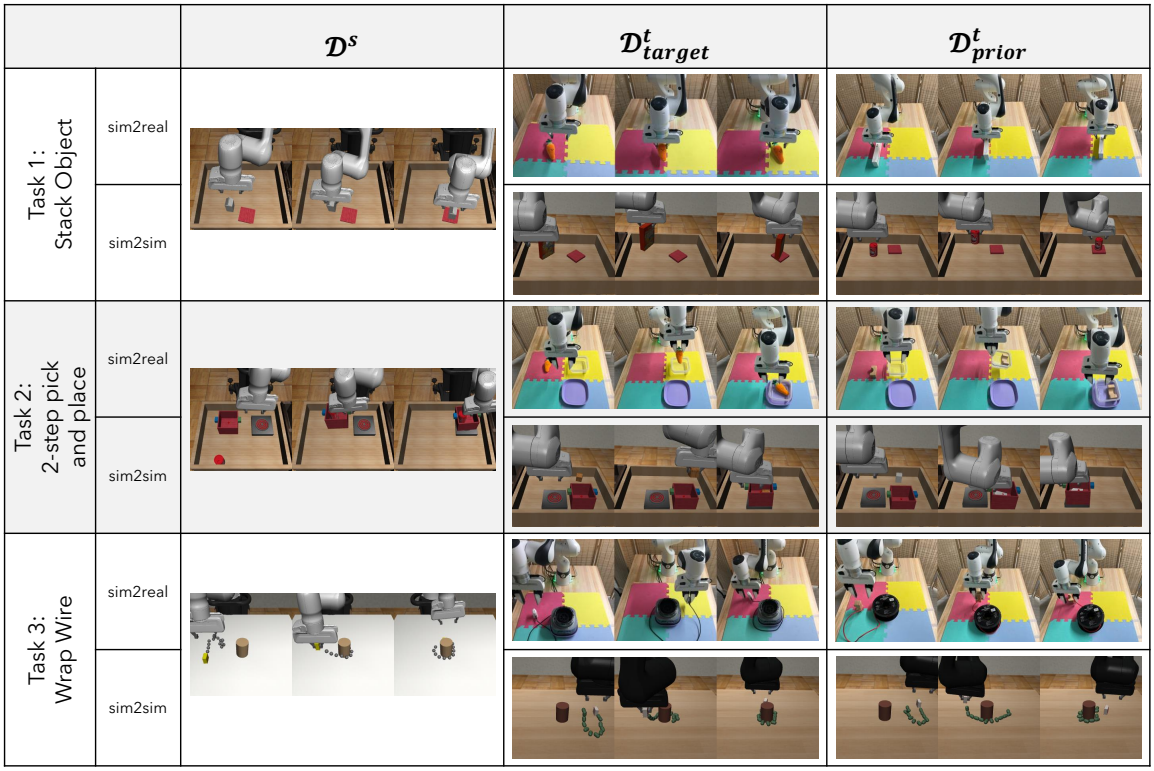


Figure 4.5: This table builds on Figure 4.4 and depicts the 3 datasets for each task with filmstrips. The rows show the three task suites while each column represents one of the three datasets we use during pretraining or policy learning. Our main results in Tables 4.4 and 4.5 use $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$ for pretraining and policy learning, whereas our results in Table 4.7 use $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{prior}}$ for pretraining and $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$ for policy learning. This table shows the visual differences between sim and real, as well as the task in $\mathcal{D}^{\text{t}}_{\text{prior}}$ versus $\mathcal{D}^{\text{t}}_{\text{target}}$.

bounds as they are given $\left|\mathcal{D}^{\text{t}}_{\text{prior}}\right| = 50$ additional action-labeled demonstrations. In other words, during policy learning, the baselines train on action-labeled demonstrations from $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{prior}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$ while ours are only trained on $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$. Results are shown in Table 4.7.

How different are $\mathcal{D}^{\text{t}}_{\text{prior}}$ ***and*** $\mathcal{D}^{\text{t}}_{\text{target}}$***?*** In `sim2sim` and `sim2real` for stack object and 2-step pick-and-place, the robot interacts with different objects in the two real-world tasks. Instead of a carrot as in $\mathcal{D}^{\text{t}}_{\text{target}}$, in $\mathcal{D}^{\text{t}}_{\text{prior}}$, the robot interacts with a paper box for the stack object task suite and a rigid toy wooden block for 2-step

Table 4.7: `sim2real`: Performance in $\mathcal{D}^{s} \cup \mathcal{D}^{t}_{target} \cup \mathcal{D}^{t}_{prior}$ setting by Number of Target Task Demonstrations

Method	Action-labeled Data			Stack Object			Multi-step Pick and Place						Wrap Wire		
	Sim	Real		Success Rate (%)			Success Rate (%)			Subtasks Completed			Success Rate (%)		
	$\mathcal{D}^{s}$	$\mathcal{D}^{t}_{target}$	$\mathcal{D}^{t}_{prior}$	25	50	100	25	50	100	25	50	100	25	50	100
No Pretrain ($\mathcal{D}^{t}$ data only)	–	✓	✓	45	30	65	40	20	30	1.15	0.9	1.15	25	45	35
No Pretrain ($\mathcal{D}^{s} + \mathcal{D}^{t}$ data)	✓	✓	✓	20	55	25	45	30	50	1.25	1.2	1.4	15	30	30
MMD	✓	✓	✓	35	30	40	70	45	35	**1.65**	1.25	1.2	15	0	20
Domain Random.	✓	✓	✓	25	45	60	15	15	20	0.9	0.55	0.85	0	5	5
ADR+RNA	✓	✓	✓	15	10	20	50	5	50	1.35	0.7	1.25	15	10	20
Lang Reg. (ours)	✓	✓	–	50	55	**85**	55	**80**	**95**	1.2	**1.8**	**1.95**	25	**50**	55
Lang Dist. (ours)	✓	✓	–	30	65	70	25	50	65	0.95	1.4	1.5	15	25	60
Stage Classif.	✓	✓	–	70	60	70	20	60	85	0.9	1.5	1.8	15	20	**70**
CLIP (frozen)	✓	✓	✓	30	25	35	25	45	35	0.55	0.95	0.95	**35**	40	45
R3M (frozen)	✓	✓	✓	**80**	**70**	80	**75**	75	85	1.6	1.55	1.75	30	25	20

pick-and-place.

In `sim2sim` on wire wrap, $\mathcal{D}^{t}_{prior}$ contains data of the beads being wrapped clockwise, instead of counterclockwise in $\mathcal{D}^{t}_{target}$. In `sim2real` for wire wrap, the plug, cord, and blender in $\mathcal{D}^{t}_{target}$ are replaced by a wooden block, ethernet cable, and spool, respectively, in $\mathcal{D}^{t}_{prior}$ data. The differences between $\mathcal{D}^{t}_{prior}$ and $\mathcal{D}^{t}_{target}$ can be visually examined in Figure 4.5.

What trends are different between Table 4.7 (with $\mathcal{D}^{t}_{prior}$) and Table 4.4 (without $\mathcal{D}^{t}_{prior}$)? Most of the trends are similar. Re-examining our main experimental questions, we see that our method still nearly doubles the success rate of both non-pretrained baselines, outperforms all three prior sim2real baselines, and that using language regression is important to achieve the most gains from pretraining (language regression outperforms stage classification and language distance, on average). However, in this new problem setting in `sim2real`, R3M outperforms our method in the lowest data regime with 25 target task demonstrations, perhaps because of the additional 50 $\mathcal{D}^{t}_{prior}$ demonstrations that our method does not train on. However, on 50 and 100 trajectories for the longer-horizon multi-step pick and place task, our method achieves higher sim2real performance than the best of either pretrained baseline.

Table 4.8: `sim2real`: Finetuning R3M with our method

Pretraining	Action-labeled Data			Multi-step Pick and Place						Wrap Wire		
	Sim	Real		Success Rate (%)			Subtasks Completed			Success Rate (%)		
	$\mathcal{D}^s$	$\mathcal{D}^t_{\text{target}}$	$\mathcal{D}^t_{\text{prior}}$	25	50	100	25	50	100	25	50	100
R3M + adapters, Lang Reg.	✓	✓	–	0	30	40	0.7	0.95	1.25	0	5	5
Lang Reg. (ours)	✓	✓	–	55	**80**	**95**	1.2	**1.8**	**1.95**	25	**50**	**55**
R3M (frozen)	✓	✓	✓	**75**	75	85	**1.6**	1.55	1.75	**30**	25	20

4.6.1.8 Can our method be combined with prior large-scale vision-language pretrained networks?

We implemented and evaluated multiple ways to combine R3M with our image-language pretraining to see if it would be possible to leverage the benefits of both R3M's large-scale pretraining and our method's domain-invariant representation learning. Instead of initializing a ResNet from scratch before image-language pretraining, we experiment with using R3M weights and finetuning the last layer, the entire network, or inserted convolutional adapter modules Chen et al. (2024a). Finetuning adapters (denoted R3M+adapters) performs the best in `sim2sim`, matching the performance of our method on 2-step pick-and-place.

Based on this sim2sim performance, we evaluated R3M+adapters on `sim2real`, but this performed worse than either frozen R3M or our method in `sim2real` (Table 4.8). Our findings are similar to Zhai et al. (2021) in that finetuning R3M on our pretraining procedure does not yield improvements on downstream policy learning. We hypothesize that this is because during image-language pretraining on both sim and real images, the trainable adapters learn to pick out features primarily in the simulation images as this is out-of-distribution for R3M which was trained on real-world videos, which enables R3M+adapters to do well in `sim2sim` but not `sim2real`.

4.6.1.9 Does language similarity imply action distribution similarity?

We hypothesize that one of the ways language is an effective bridge for sim2real transfer is that the sim and real action distributions of the demonstrations are simi-

lar when the image observations have similar language descriptions. Figure 4.6 shows the action distribution similarities between sim and real when the language descriptions are similar (top row), and when the language descriptions are different (bottom row). Each column represents a component of the action distribution. We plot three components: z-axis actions, xy-magnitude (which is the $\ell 2$ norm of the (x, y) action dimensions), and the gripper dimension. We observe that action distributions are indeed more similar for images described by similar language than for images described by different language.

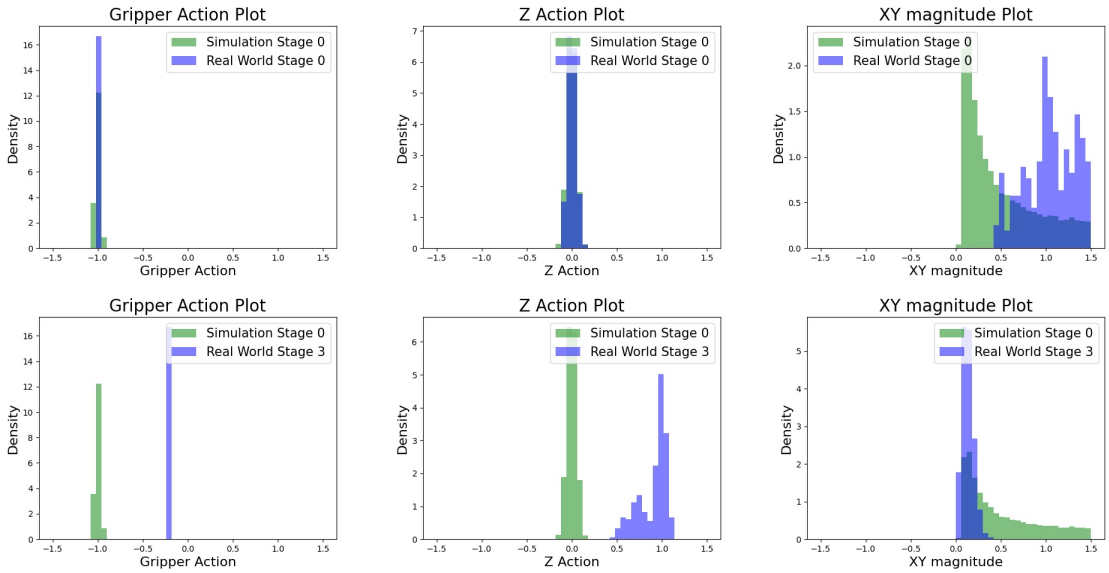


Figure 4.6: These plots show the action distribution of demonstrations across both sim and real, broken down by each component of the action: xy-action magnitude, z-axis actions, and gripper actions. The first row shows simulation (green) and real world (blue) action distributions for images described by similar language. The second row shows the same distribution of simulation actions (green) as in the first row, but compared with real-world action distributions from images labeled with very different language from the sim actions (blue). Notably, the action distributions are generally similar for images with similar language (first row), and different for images with different language (second row). This suggests that pretraining our CNN on language embedding prediction benefits downstream policy learning because it allows the domain-invariant learned representations to tap into similar action distributions for completing a task.

4.6.1.10 What are some failure modes of our policy?

See our project website (`https://robin-lab.cs.utexas.edu/lang4sim2real/`) for failure videos. In **stack object**, most failures arose from not closing the gripper enough before attempting to lift the object, since our gripper action dimension is analog (controls the distance between the gripper fingers) instead of binary (open/closed). In **multi-step pick-and-place**, most failures of our policy came from the robot not moving right after successfully completing the first pick-and-place operation. In **wrap wire**, most failures were from grasping imprecision, since the small plug allowed for only a ≤ 3 cm xy-plane error in the grasping point. Other common failures included prematurely opening the gripper and dropping the plug before the wrapping was finished, or moving the gripper sideways into the blender instead of properly circling around it.

4.7 Conclusion

Vision-based policies struggle with distributional shift during sim2real transfer. To address this challenge, we introduced a low-data-regime visual pretraining approach that leverages language to bridge the sim2real visual gap with only 25-100 real-world trajectories with automatically generated language labels. We evaluate the effectiveness of our approach on multi-step long-horizon tasks and hard-to-simulate deformable objects. In the few-shot setting, our approach outperforms state-of-the-art vision-language foundation models and prior sim2real approaches across 3 task suites in both `sim2sim` and `sim2real`.

4.8 Limitations and Future Work

One of the main limitations of our work is that the learned representation may have limited generalizability compared to existing pretraining methods that leverage internet-scale data to enable a large degree of generalization. Our approach targets

a specific distribution and domain of real-world tasks and operates in the low-data regime for both pretraining and policy learning, so it does not yield general-purpose visual representations that can be applied to a wide distribution of target tasks. Future work could investigate scaling our method to large-scale datasets to further reduce the number of real-world demonstrations needed for effective sim2real transfer.

Our method also assumes that the template language descriptions used by the automatic labeling process describe similar aspects of images across the two domains, and may perform worse if the language between sim and real described images at extremely different levels of granularity. Furthermore, our approach relies on segmenting all trajectories of a task into stages of a certain granularity so that the associated template language is diverse enough to prevent the learned visual representation from mapping the entire input image distribution to a collapsed point. On contact-rich tasks involving continuous motions or complex object deformations, it may be harder to segment a trajectory and label these segments with language.

Another avenue for future work involves exploring sim2real by combining existing pretraining approaches such as time-contrastive learning and masked image modeling in conjunction with the language-based pretraining we propose, as adding temporal or masked prediction terms to the objective may enable more fine-grained representations that complement the coarseness of language.

Chapter 5: Mixed-Initiative Dialog for Human-Robot Collaborative Manipulation

In Chapters 3 and 4, we saw how language, as a store of semantic meaning, enables a surprising amount of few-shot generalization to new tasks *and* new domains. In this chapter (based on work published at ICRA 2026 (Yu et al., 2025)), we pivot to our second line of work mentioned in Section 1.3 to explore how language can enable flexible communication paradigms for human-robot collaboration. Why is collaboration important, and how can effective communication expand physical robotic capabilities?

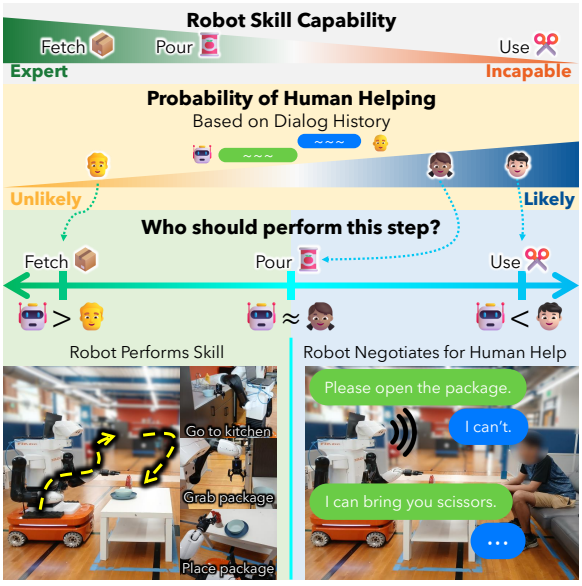


Figure 5.1: We present MICoBot, a system for human-robot collaboration where both agents can initiate and carry out physical and verbal actions. MICoBot uses both the robot's capability and the likelihood of human helping (inferred from previous dialog history) to determine whether the robot is better suited than the human to perform the skill. If it is, it attempts the skill itself. If not, it negotiates for human help.

5.1 Introduction

Imagine preparing for a dinner party with a friend. Your friend might excel at mixing drinks while you focus on cooking the main dish. You are also better at decorating, while both of you reluctantly negotiate over less desirable tasks like cleaning.

Now, imagine a helper robot in place of the friend. Current robots are not fully autonomous for many household tasks, but they offer broad capabilities with varying levels of reliability that can be leveraged through collaboration with humans.

To be an effective partner, such a robot must communicate in physically grounded natural language, decide when to take initiative or defer to the human, negotiate task allocation based on strengths and preferences, and adapt to changing contexts. These ingredients are essential not only for collaborative household robots, but also for coding assistants, chatbots, and AI agents more broadly.

Long-horizon tasks, such as preparing for a party, require dynamic, bidirectional collaboration across control, initiative, and communication. In particular, the ability to both take initiative and yield control is central to effective human–AI teamwork. However, current AI systems (e.g., chatbots) typically rely on one-directional, human-initiated interactions (Ouyang et al., 2022; Achiam et al., 2023), while prior human–robot interaction (HRI) approaches often assume fixed collaboration plans and full human compliance (Selvaggio et al., 2021). Such assumptions limit flexibility and fail to account for the diverse preferences, capabilities, and strengths of different human partners. We argue that effective human–robot collaboration requires a paradigm shift toward mixed-initiative dialog as the communicative medium, enabling both agents to initiate, negotiate, and respond to proposals in natural language.

To enable this paradigm shift, we introduce MICoBot (Mixed-Initiative Collaborative roBot), the first system that supports mixed-initiative dialog for seamless human–robot collaboration in the physical world. MICoBot allocates task steps to the most suitable agent (see Fig. 5.1) in a way that maximizes overall success, minimizes human effort, and respects human-initiated requests. It achieves this by engaging in mixed-initiative dialog and negotiation to decide step allocation (see Fig. 5.2), while coordinating the physical and verbal actions required to execute the plan.

To realize this objective across diverse humans and long-horizon tasks, MICoBot optimizes decision-making at three levels. First, a meta-planner determines the high-level collaboration strategy, integrates human-specified preferences, and generates adaptive code for robot actions (verbal or physical). Second, a planner executes this code, selecting the optimal collaboration approach based on the environment

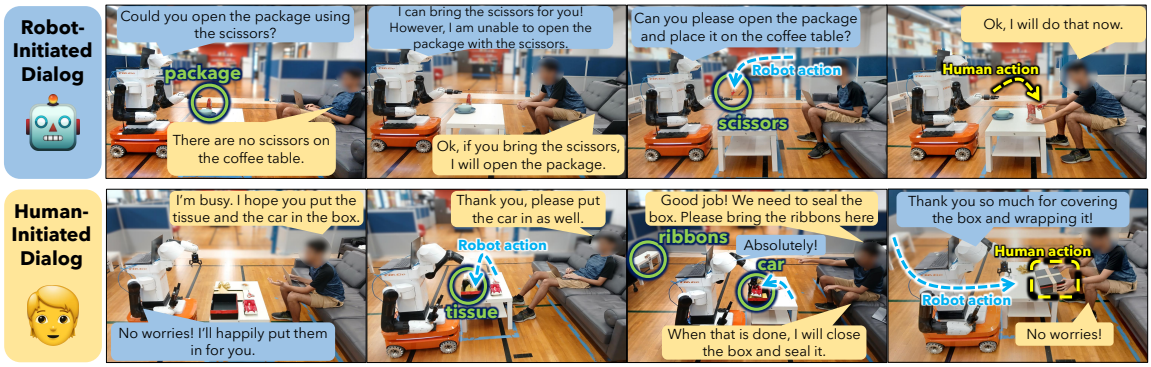


Figure 5.2: MICoBot supports *both* robot-initiated (**top row**) *and* human-initiated (**bottom row**) task-directed speech2speech dialog, where both agents discuss who is best suited to perform steps in a long-horizon task. These are real dialog and physical interactions from our user studies (see our project site at `https://robin-lab.cs.utexas.edu/MicoBot/`).

state, a simulation-trained affordance model of robot capabilities, and a dynamic estimate of human helpfulness derived from prior interactions. Finally, an action executor carries out the next step of the plan, whether it involves manipulation, dialog initiation, or dialog response.

We validate MICoBot through extensive evaluation in both simulation and the real world. In simulation, we test against LLM-simulated humans with varying helpfulness and responsiveness; in real-world experiments, 18 participants collaborated with a TIAGo mobile manipulator on three household tasks. Our approach improves success rate by **50%** compared to a pure LLM baseline and is preferred by over **75%** of participants.

In summary, our contributions are:

1. **A new problem setting** that integrates mixed-initiative natural language dialog with mixed-initiative human-robot interaction.

2. **A novel optimization framework** for task allocation that balances human and robot effort with success through a unified metric.

3. **A collaborative simulation environment** built on MiniBehavior (Jin et al., 2023), featuring LLM-controlled virtual humans, with an interactive demo on our project site (`https://robin-lab.cs.utexas.edu/MicoBot/`).

4. **A hierarchical robotic system, MICoBot**, that enables mixed-initiative speech-to-speech human-robot collaboration and flexibly adapts to diverse real human collaborators in physically grounded, long-horizon tasks.

5.2 Related Work

5.2.1 Mixed-initiative Dialog

Mixed-initiative dialog (Carbonell, 1970; Allen et al., 1999; Chu-Carroll, 2000) refers to communication with freeflowing questions and answers from both parties. In the NLP field, the dominant chatbot paradigm adopted by large language models (LLMs) largely eschews mixed-initiative interaction: humans pose substantive questions, and the chatbot primarily responds to fulfill these requests (Ouyang et al., 2022; Achiam et al., 2023). Recent work has sought to make dialog systems more goal-directed and proactive by incorporating mixed-initiative strategies in tasks such as creating documents (Wu et al., 2025), persuading users to donate to charity, enhancing users' emotional well-being (Deng et al., 2023a; Yu et al., 2023b; Chen et al., 2023; Deng et al., 2024), clarifying ambiguous human requests (Qian et al., 2021; Deng et al., 2023b; Chen et al., 2024c), or as part of an active-learning framework (Thomason et al., 2019b). However, none of these systems addressed mixed-initiative dialog in grounded, real-world collaborative scenarios involving physical manipulation tasks.

5.2.2 Human-Robot Collaboration with Dialog

In the human-robot interaction (HRI) field, researchers have developed human-robot collaboration systems that interact through language but are restricted to **single-initiative dialog**. Some of these systems integrate LLMs as task planners or delegators (Wang et al., 2024; Mandi et al., 2023; Feng et al., 2024) for tasks

like real-world cooking (Wang et al., 2024) and object sorting (Mandi et al., 2023). Other systems implement a leader-follower paradigm in simulated worlds, where the leader issues natural language instructions for the follower to execute (Suhr et al., 2022; Kojima et al., 2021; Team et al., 2022; Gao et al., 2023). Single-initiative HRI systems can ask humans for clarification (Ren et al., 2023) or assistance (Bennetot et al., 2020; Knepper et al., 2013; Veloso et al., 2015), or inform humans of their observations (Chen et al., 2010; Mutlu et al., 2006; Cascianelli et al., 2018). However, by supporting only single-initiative dialog, these systems lack the capacity to adapt to the evolving nature of the human, robot, and environment—limiting their capacity to find the optimal division of labor that respects user preferences (Mandi et al., 2023).

5.2.3 Mixed-initiative Collaborative Systems Without Dialog

Some works in HRI have explored mixed-initiative collaborative systems without dialog, only with physical actions (Few et al., 2006; Natarajan et al., 2024; Bishop et al., 2020; Rosero et al., 2021; Paleja et al., 2024; Jiang and Arkin, 2015). In particular, (Baraglia et al., 2016) studied separate regimes of agent initiative (human-initiative: requesting help, or robot-initiative: proactively helping), but failed to support a natural human-robot dialog. By focusing solely on physical actions, these prior works overlook the critical role of communication in effective collaboration, thereby limiting the flexibility of the human-robot team. With MICoBot, we enable both agents to take initiative—through both physical and verbal actions—via task-grounded dialog.

5.2.4 Human-Robot Task Allocation

Several prior works in robotics and planning have studied the problem of human-robot optimal task allocation, typically optimizing the time to perform a task or minimizing idle agents, posing the problem as a scheduling optimization (Vats et al., 2022; Yu et al., 2021a). Others have prioritized different objectives, such as

safety (Faccio et al., 2024) through the formulation of a constrained optimization problem (Singh et al., 2023). While these solutions may achieve shorter execution times, they assume a priori known capabilities and availability of all agents, including both robots and humans. In contrast, MICoBot can adapt to the specific human's willingness to help by estimating its availability based on previous dialog.

5.2.5 Agents with Both Physical and Verbal Action Spaces

MICoBot relies on a heterogeneous action space that includes interacting with the physical world *and* generating freeform dialogue to a human collaborator. Prior works have developed policies with a combined physical and verbal action space through RL (Das et al., 2017; Shervedani et al., 2024) or IL (imitation learning) (Padmakumar et al., 2022; Team et al., 2022). Research on language emergence in multiagent systems (Peters et al., 2025; Kottur et al., 2017) has also examined how cooperative agents learn to communicate through latent representations or natural language when performing simulated robotic tasks (Lin et al., 2021; Lowe et al., 2020; Woodward et al., 2019; Lobos-Tsunekawa et al., 2022; Kolb et al., 2019). However, these works are typically limited to simulated domains, where action spaces and task dynamics are highly abstracted or simplified. They often rely on limited communication protocols without integrating grounded task structure, rich human preferences, or real-world execution constraints. In contrast, MICoBot leverages an LLM to generate freeform, grounded dialog within a shared task context, enabling fluid mixed-initiative interaction and reasoning over both verbal and physical actions in real-world scenarios.

5.2.6 Natural Language and Robotics

Our work sits at the broad, growing intersection of natural language and robot learning. We refer the reader to various lines of work upon which different modules of our method are based, including language-conditioned robot policies (Jang et al.,

2021; Lynch and Sermanet, 2021; Mees et al., 2021, 2022; Shao et al., 2020; Sodhani et al., 2021; Silva et al., 2021; Karamcheti et al., 2021; Jiang et al., 2022; Shah et al., 2023; Yu and Mooney, 2022), LLMs as task planners (Huang et al., 2022; Ahn et al., 2022; Chen et al., 2022; Raman et al., 2023), code-based policies (Liang et al., 2022; Li et al., 2024b; Huang et al., 2023), hierarchical policies (Shi et al., 2025, 2024; Belkhale et al., 2024) and planners (Choi et al., 2025; Luo et al., 2023), vision-language representations (Radford et al., 2021; Zhai et al., 2021; Zhu et al., 2023) for robotic control (Nair et al., 2022; Shridhar et al., 2021, 2022; Yu et al., 2024), and language-based reward shaping for RL policies (Nair et al., 2021b; Goyal et al., 2019, 2020; Fan et al., 2022; Ma et al., 2023, 2024a,b; Yu et al., 2023a).

5.3 Problem Setting: Task Collaboration with Mixed-Initiative Dialog

5.3.1 MDP Formulation

We study how human-robot collaborative manipulation can be facilitated through mixed-initiative dialog. We assume that both agents can observe the state of the world, $s \in \mathcal{S}$, and perform actions, $a \in \mathcal{A} = \mathcal{A}_p \cup \mathcal{A}_v$, comprised of a physical action space, $\mathcal{A}_p$ (e.g., move objects, open them, etc.), that directly affect the physical state of the environment s, and a free-form, natural language verbal action space, $\mathcal{A}_v$, which is directly observed by the other agent but does not change the physical state. We model the problem as a Markov Decision Process (MDP) from the robot's point of view (see Fig. 5.3), where on each environment step, the robot performs some action, $a_R \in \mathcal{A}_{p,R} \cup \mathcal{A}_{v,R}$ and receives an observation $o = [I, a_{v,H}, s_{proprio}]$ consisting of an

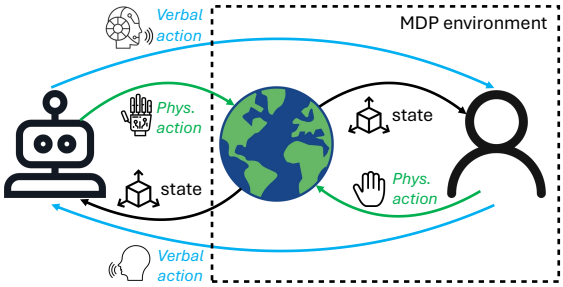


Figure 5.3: Proposed MDP for Mixed-Initiative Collaboration.

RGB-D image I, an optional verbal action from the human partner $a_{v,H}$, and the robot's proprioceptive state $s_{proprio}$. Within each environment step, the human may perform a series of actions, $a_H \in \mathcal{A}_{p,H} \cup \mathcal{A}_{v,H}$, in its own physical and verbal action space after perceiving the world state and robot's previous dialog, $a_{v,R}$.

5.3.2 Physical and Verbal Action Spaces

The physical and verbal action spaces, $\mathcal{A}_p$ and $\mathcal{A}_v$, are shared between both agents. Each element of these action spaces is a parameterized action primitive represented by the pair, $a_{p/v} = (\omega_{p/v}, \theta_{p/v})$. ω_p is the type of the physical action primitive (`open`, `pick-and-place`, etc.) and θ_p are the corresponding parameters (e.g., what object to open or pick and where to place it). We assume that humans are fully competent in executing all steps of a collaborative household manipulation task but may be unwilling or unavailable to perform some or all required actions. Their behavior can range from indifferent (never acting) to overly proactive (completing the entire task without robot involvement).

In contrast, robots often have limited manipulation capabilities and may be unable to execute more complex actions, in which case it uses verbal actions to communicate with the human. ω_v is the type of the verbal action primitive (`ask_human_for_help`, `respond_to_human`, etc.), and θ_v are the corresponding parameters defining the context of the verbal primitive (e.g., what step the robot needs help on). While the types of verbal actions are limited, each generates freeform and open-vocabulary language. MICoBot first selects an abstract verbal action from this space, then translates it into a natural language utterance to negotiate with the human—conveying its requests and the assistance it requires for successful collaboration. This involves reasoning over asymmetric human and robot physical capabilities to devise collaboration strategies that maximize task success while minimizing human effort.

5.3.3 Collaborative Task Definition and Problem Statement

We assume the collaborative task is defined by a task plan of length T, known to both agents and represented as a sequence of unassigned **physical** action primitives, $[a_{p,0}, ..., a_{p,T-1}]$, such as [(`pick-and-place`(`box`, `table`), ..., `close`(`box`)], obtained from the task instructions or off-the-shelf task planner. To complete the manipulation task while minimizing human effort, the system must allocate steps of the plan between the two agents—negotiating with the human through robot-initiated dialog to suggest assignments, adapting to human preferences through human-initiated dialog, and ultimately executing its assigned physical actions. At each step t, the system must compute the best allocation of the remaining steps of the plan, $G = [g_t, ..., g_{T-1}]$, where $\forall t, g_t \in \{H, R\}$. The optimal allocation G^* maximizes the expected task success probability while minimizing total human effort. These objectives are inherently competing: a policy focused solely on maximizing success might allocate all steps to the human (assumed to be perfectly competent); conversely, minimizing human effort alone would assign all steps to the robot, even when it may be incapable of completing certain steps. The optimization also incorporates constraints conveyed through the mixed-initiative dialog history, such as task allocation requests or proposed task splits. The resulting allocation G^* determines whether the robot executes the current step (R) or negotiates with the human for assistance (H).

5.4 MICoBot: Mixed-Initiative Collaborative Robot

5.4.1 Collaborative Task Allocation as Optimization.

A helpful physical collaborator must aim for task success with minimal human effort while adhering to human preferences expressed in dialog (i.e., for certain steps to be done by a certain agent). Therefore, we formulate our objective for collaborative task allocation as a constrained optimization problem, where constraints are updated based on dialog exchanges. To avoid a complex multi-objective formulation, we combine success probability and effort into a single Q-value by building on prior

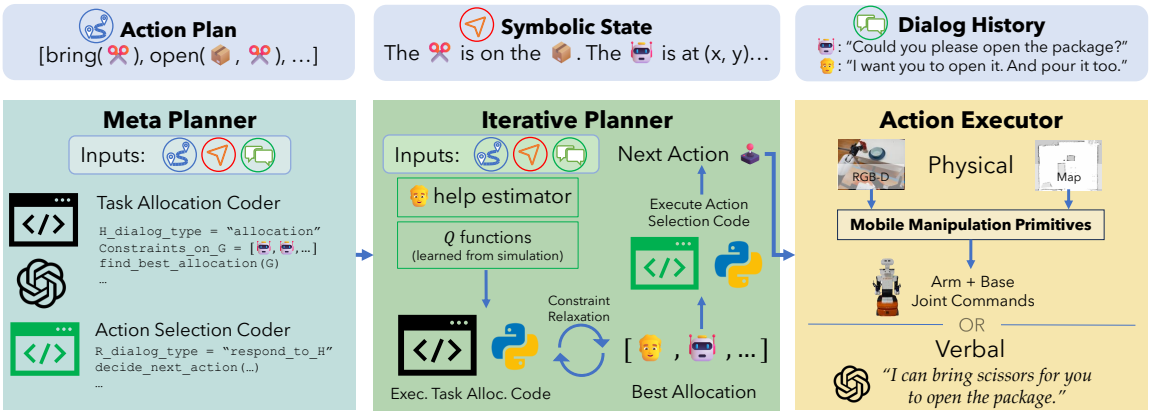


Figure 5.4: MICoBot consists of 3 decision-making modules: a meta-planner that produces a collaborative strategy expressed through adaptive planning code, a planner that executes the code and optimizes our objective (Eq. 5.1) to decide the next primitive action, and an action executor that outputs the low-level pose trajectory or verbal utterance to say to the human.

work on temporal distances in RL (Myers et al., 2025). Then, to allocate task steps, MICoBot compares robot and human Q-values.

We assume each task step is executed by a multi-task policy π that performs continuous low-level control at a fixed control frequency. In this low-level MDP (distinct from the high-level task MDP described in Sec. 5.3), we define the reward as $r = -1$ per time step until the skill completes or times out, at which point $r_{termination} = 0$. A well-trained Q-function, $Q : o_t \times a_t = (\omega_t, \theta_t) \mapsto \mathbb{R}$ with a discount factor of 1, thus represents the **negative expected number of timesteps** until skill completion from a given state. For a perfectly competent agent (i.e., human), this corresponds to the average timesteps required to perform the action. For an imperfect agent that may fail, the Q-function reflects a weighted expectation over both successful and failed outcomes—where failure contributes a significant timestep penalty (timeout) weighted by its probability. We assign each agent a distinct Q-function: Q_R for the robot and Q_H for the human. These agent-specific Q-functions thus provide a unified, interpretable cost metric for comparing step allocations, jointly capturing both execution time (effort) and likelihood of success.

However, directly optimizing step allocation using only these two Q-functions diverges from realistic human-robot collaboration scenarios in three ways: (1) human and robot effort are valued equally, ignoring the higher worth of human time and attention; (2) the human is assumed to always comply with robot-initiated requests, overlooking variability in willingness or availability; and (3) human-initiated requests or preferences are not considered, limiting the system's adaptability to human intent. To address (1), we introduce a *human-effort factor*, α, a ratio valuing human effort to robot effort. To address (2), human Q-values are adjusted with an inferred probability, $p_{H,t}$, of the human agreeing to perform action $a_{H,t} = \omega_t(\theta_t)$ when asked. For less cooperative users, this probability lowers the expected success of $a_{H,t}$, effectively increasing the magnitude of the negative Q-value due to potential human refusal. To address (3), we enforce constraints, $C_1, \ldots, C_n$, extracted from human-initiated dialog—such as explicit requests to perform specific steps themselves or delegate them to the robot. Altogether, we propose the following objective to find the optimal task allocation G^*:

$$\begin{aligned} \max_{g_t,\ldots,g_T} \quad & \sum_{t}^{T-1} \left(\mathbb{1}_{g_t=H} \cdot \frac{\alpha}{p_{H,t}} + \mathbb{1}_{g_t=R} \right) Q_{g_t}(s_t, a_t), \\ s.t. \quad & C_1, \ldots, C_n \text{ are satisfied} \end{aligned} \tag{5.1}$$

that minimizes expected time-to-success and human effort.

5.4.2 MICoBot Framework

MICoBot is a three-level framework (Fig. 5.4) that includes 1) a **meta-planner** that processes human dialog and generates a collaborative strategy expressed in code, 2) an **iterative planner** that updates planning state variables and allocates and decides the next action to perform by executing the code, and 3) an **action executor** that carries out the action primitive, either through low-level physical actions or by formulating a dialog utterance to communicate to the human.

5.4.2.1 L1: Meta-planner

The meta-planner produces adaptive planning code that dictates the overall strategy for L2 and L3 to follow. Based on the most recent human dialog, the current symbolic state of the world, the task plan, and approximately 15 in-context learning (ICL) examples, it generates two pieces of code: first, **task allocation** code to adapt the optimization computation, such as to map human dialog into additional constraints, and second, **action selection** code for how to choose the next action, such as whether to engage in additional dialog, proposals to split up the task with the human, or negotiation rounds before proceeding further with physical steps in the plan. The meta-planner is implemented as an LLM-based (GPT-4o) coder, and the prompt can be found on our project website (`https://robin-lab.cs.utexas.edu/MicoBot/static/txts/metaplanner_prompt.txt`).

Fault Recovery. The metaplanner occasionally produces faulty, non-executable code. For fault recovery, the metaplanner is automatically re-queried up to 2 additional times to create code. If these attempts also produce non-executable code, the most recent dialog from the human is ignored for 2 further, automated metaplanner requeries. These re-queries are handled by a try-except block in the iterative planner module.

5.4.2.2 L2: Iterative Planner

The iterative planner runs code from the meta-planner (L1) to make two key decisions: whether to initiate dialog and which verbal or physical action to perform. L2 runs in two stages. *First*, it performs constrained optimization to find the best task allocation for the remaining task steps. To do this, we evaluate Eq. 5.1 under all possible task allocations fulfilling the constraints. Initially, the planner attempts to incorporate all constraints from the mixed-initiative dialog history. If no feasible allocation is found (e.g., human asked the robot perform a step it is incapable of), the planner iteratively relaxes the most recent constraint, and the robot verbally explains

its incapability.

To evaluate Eq. 5.1, MICoBot requires accurate Q-functions that capture each agent's expected effort and success probability on each task step. To collect data to learn the robot's Q-function (Q_R), we use the OmniGibson simulator (Li et al., 2022), configured with a coarse model of the real-world task, environment, and action primitives, recording both completion times and success rate. We train a supervised network as Q_R that predicts the expected timesteps for an action primitive a to succeed from a given symbolic state o. Conditioning Q_R on symbolic states minimizes the sim-to-real gap when we deploy the function to our real-world setting. When estimating the human's Q-function (Q_H), we assume perfect competence (i.e., no execution failures). Thus, we simply obtain time estimates for each step from an LLM predicting how long a human needs to execute action $a_t = \omega_t(\theta_t)$, plus a travel time estimate based on human-object distances.

To adapt to changing human helpfulness, MICoBot estimates the probability of human assistance at the current t-th timestep, $p_{H,t}$, using an LLM-based sentiment analysis over prior human-robot dialog (see prompt on our project website at `https://robin-lab.cs.utexas.edu/MicoBot/static/txts/p_help_estimator_prompt.txt`). This enables MICoBot to adapt to temporally-changing user sentiments. After deciding the optimal task allocation, the second stage of L2 executes meta-planner action selection code to generate the optimal action $a = \omega(\theta)$ to execute: a physical mobile manipulation primitive ω to perform a task step on objects specified in θ, or a verbal primitive ω to initiate dialog to ask for help, propose splitting up steps, or respond to human-initiated dialog regarding specific task steps specified in θ.

5.4.2.3 L3: Action Executor

The action executor performs the action primitive selected by the planner (L2). For *physical actions*, it generates a trajectory for navigation and arm movement to reach the location of and manipulate the target object while avoiding ob-

stacles. We build a pipeline similar to Shah et al. (2025) that uses the `move_base` ROS package for navigation path planning over a 2D occupancy map, and Grounding DINO (Liu et al., 2023b) to segment the target object from an open-world scene based on the object specified in θ. We backproject segmented image pixels from RGB-D camera data into a 3D point cloud to identify graspable or placeable points in the robot's workspace that the arm reaches through inverse kinematics (IK). For *verbal actions*, an LLM generates free-form natural language utterances to communicate with the human based on the dialog intent ω and verbal action parameters θ decided in L2. The LLM uses 10 in-context-learning examples (see prompt on our project website at `https://robin-lab.cs.utexas.edu/MicoBot/static/txts/dialog_action_executor_prompt.txt`) to generate language grounded in the context of the task and dialog.

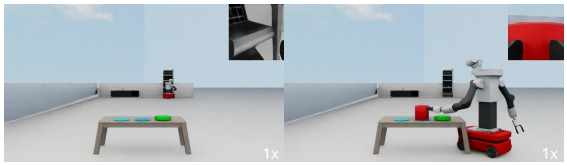
5.4.3 Hierarchical Plan

To streamline communication for long-horizon task plans, MICoBot groups adjacent low-level steps into semantically meaningful abstract actions that can be discussed more succinctly with the human. The system only descends to a finer-grained level of detail during negotiation over low-level step assignments, which reduces the frequency and complexity of dialog, resulting in more efficient and user-friendly communication.

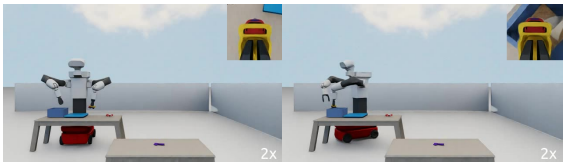
5.4.4 L2: Iterative Planner Component Details

5.4.4.1 Robot Q-function Q_R training in OmniGibson

To train Q-functions for the robot, we first create a simulated OmniGibson environment with a PAL Tiago robot and an environment that roughly matches the relative locations of the relevant furnitures and objects. We then implement each real-world skill first in OmniGibson. Fig. 5.5 depicts example frames from primitives in task 1 and task 3 we run in the OmniGibson simulator to collect sample Q-values



Task 1, Step 2: Pick package from shelf (left) and place on coffee table (right).



Task 3, Step 3: Pick toy car from coffee table (left) and place into gift box (right).

Figure 5.5: Frames from primitive rollouts in OmniGibson for task 1 (left two images) and task 3 (right two images). Left and right images for each task capture early and late frames of the skill. The square image at the top right of each frame represents the robot's camera view observation.

for each skill.

We collect samples of the form $(o, a, \mathcal{T})$, where o is the initial observation of the world, a is the skill-parameter pair (ω, θ) taken by the robot at o, and $\mathcal{T}$ is the number of timesteps the robot takes to succeed at a from o. If the robot does not succeed in its execution, then $\mathcal{T}$ is set to some fixed constant representing the maximum number of timesteps allowed in each skill-parameter execution.

To train our Q-functions, we collect roughly 100 samples for each action a and train with inputs (o, a) and target Q-values $-\mathcal{T}$ using ℓ_2 regression with the Adam Optimizer. Since our observations o are primarily symbolic but include some positional information of the robot and objects, our network architecture is extremely lightweight—2 linear layers with hidden size 32 and an output size of dimension 1 for the Q-value.

5.4.4.2 Human Q-function Q_H Estimation

To estimate Q_H, we compute two terms. The first is the human's stationary cost—the number of seconds for the human to perform some task if the relevant items were all right in front of them, within grasp. This term was copied from the output of an LLM call, which was prompted with a natural language description of the low-level step in the task alongside a URL to a product description of the toy car (for

task 2). The second term is the human's traveling time—the number of seconds for the human to move from their current location to the object location. This was a simple 2D Euclidean distance (in meters) between the assumed human location on the couch (in the real-world user studies) and the location of the objects, divided by the average human walking speed of $1.4m/s$. We recognize this is a crude estimate of human effort, and we discuss the limitations of this in Section 5.8.

5.4.4.3 Forward Dynamics Model

Our Q-functions rely on state and action inputs. However, computing the best task allocation involves considering Q-values for future steps, which depends on having knowledge of what the future state at that step will be. This involves creating a forward dynamics model so that we can estimate the future state n plan steps into the future, which can be difficult to learn accurately for continuous states. We sidestep this problem by using symbolic states for our Q-values trained in simulation and maintaining these symbolic states during our real-world experiments. A symbolic state-based forward model is feasible to hardcode in our problem setting because we assume that each action affecting change in the world is a skill-parameter physical primitive, where the effect is quite easy to specify symbolically. For instance, the effect of `pickplace(bowl, coffee_table)` is that the bowl moves from its original furniture to the coffee table. Though this is a limitation of our method, learning a forward dynamics model is not a contribution of our work, so we leave the extension of our approach to continuous state representations to future work.

5.5 Experimental Setup

We evaluate MICoBot in the real-world, on a Tiago mobile manipulator working with 18 unique human participants on household tasks, and in simulation, on a collaborative framework we developed atop Mini-Behavior gridworld (Jin et al., 2023). In our simulation framework, a robotic agent collaborates with a simulated human with

parametrizable helpfulness and mood-varying dialog, which allows for larger-scale experimentation and controlled comparisons across methods across a wider range of human behavior and dialog dynamics. A successful robotic collaborator must achieve task success (our primary evaluation metric) while minimizing human effort (our secondary metric). We also report **subjective measures of robot behavior**, including user satisfaction, preference rankings, and Likert-scale ratings.

5.5.1 Environment and User Study Protocol

In the **real-world**, we perform our experiments in a mock apartment with a kitchen and living room area with commonplace furniture. In all of our tasks, the robot and human work together on opposite sides of a coffee table. Simulating a household setting, the participant spends nearly all of their time on the couch, where they can do their personal (i.e., non-task-related) work. The human can be as inactive or proactive as they wish in performing physical and verbal actions as defined in Section 5.3 (though we continue the trial if they initiate dialog beyond the scope). The exact instructions we read to human participants can be found in Appendix C.2.

Each human user study consists of two 20-30 minute trials, in which they collaborate with both our method and a pure LLM baseline, ordered randomly. All trials **terminate** under any of the following conditions: a primitive fails irrecoverably, $4T$ steps have elapsed for a plan of length T, an infeasible step is allocated to the robot twice consecutively, or the human refuses twice to perform a step infeasible for the robot.

In **simulation**, we ran our method, the three baselines (RL, LLM, random), and our method's four ablations (no $p_{H,t}$ estimation, no plan hierarchy, no R-initiative dialog, and no H-initiative dialog) on eight different settings of parameterized humans in simulation. These eight settings were a cross product of 2 dialog mood settings (positive and negative) and 4 ground-truth $\tilde{p}_{H,t} \in \{0.0, 0.3, 0.7, 1.0\}$ settings (where the $\tilde{p}$ denotes the ground truth probability while the plain p denotes our estimate).

10 trials were run for each method in each of the eight settings for the parameterized human.

5.5.2 Skills

To perform long-horizon household tasks, the robot has access to several mobile-manipulation action primitives. `pick_place_mobile(obj, place_loc)` moves to `obj` and places it atop `place_loc`, another object in a potentially different room. `pour(obj, cont)` travels to `obj` and pours its contents into `cont`. Finally, `fold(obj)` folds down box flaps.

To initiate and respond within mixed-initiative dialog, the robot uses the following open-vocabulary verbal action primitives for dynamic collaboration with the human: `ask_for_human_help` on a `step` the human is best suited to perform, `propose_split` to split `steps` with the human, `explain_incapability` to ask the human to perform a `step` that the robot can't perform, and `respond_to_human` to `accept/reject` requests the robot is capable/incapable of executing.

5.5.3 Baselines

Because multiple components of our method are powered by LLMs, we compare our approach to a pure LLM baseline (**LLM**) given the same information as our meta-planner: symbolic state, dialog history, task plan, and α human-robot effort tradeoff factor. The LLM baseline is also provided with a list of the robot's available skills and assumes that the human always successfully completes a step once they agree to perform it. The LLM baseline is prompted to produce a plan allocation G that primarily optimizes for task success and secondarily minimizes human effort. See our project site for the prompt for this LLM baseline (`https://robin-lab.cs.utexas.edu/MicoBot/static/txts/llm_baseline_prompt.txt`).

To control for the amount of human effort in the user studies, we compute an additional random allocation baseline that does not involve a human participant,

RECB (random effort-controlled baseline). Let p_c denote the proportion of steps done by the human in our method's trials. RECB randomly allocates the current step to the human with probability p_c and assumes a perfectly helpful human and oracle robot primitives with 100% success rate.

In simulation, we additionally compare against a naive **Random** baseline allocating either agent (with probability 50%) to perform the next step, and an **RL** baseline (hierarchical task allocator + robot policy). For this RL baseline, we train a hierarchical policy where the high-level policy was a task allocator that outputted logits over two classes: 0 (Robot would perform current step), or 1 (Human would perform current step). If the logit for 0 is higher, then the image observation is passed into the low-level robot policy that decides the discrete physical action to take in the world. Otherwise, the robot asks the human through a verbal action for help on that step. We use sparse rewards, issued only when all 5 steps were completed in the task, in the proper order. We initially trained the RL policy on two simulated human settings: one where the human ground truth likeliness of helping, $\tilde{p}_{H,t} = 1.0$, and another where $\tilde{p}_{H,t} \sim U[0, 1]$. We were unable to obtain policies with any non-zero training returns after thousands of iterations on the latter setting, so we only report results on the former setting, which explains why the RL policy does not perform well when $\tilde{p}_{H,t}$ is low.

5.5.4 Ablations

To measure the importance of mixed-initiative dialog, we perform the following ablations in simulation: **H-init** and **R-init**, where the human or the robot alone, respectively, can initiate any dialog. We further ablate components of MICoBot in simulation by running it **w/o P_H** (no $p_{H,t}$ estimation) and **w/o Plan Hierarchy** (where our method talks to the human in terms of granular, low-level steps instead of more understandable, high-level subtasks).

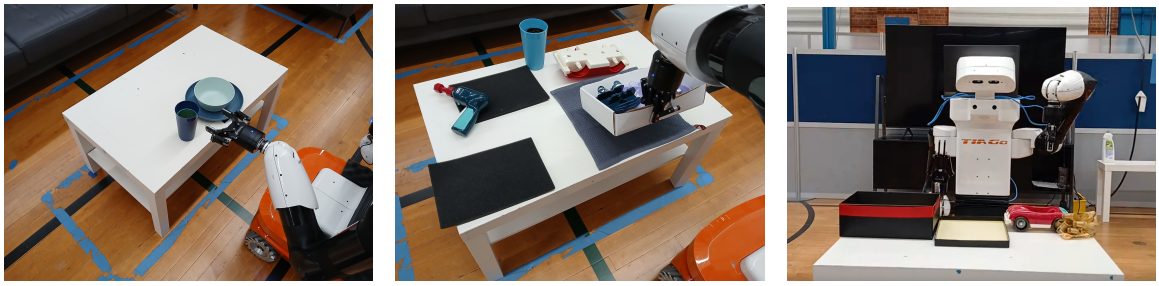


Figure 5.6: Real-world tasks from left to right: pouring package into bowl, assembling toy car, and packing gift box.

5.5.5 Metrics

Success Rate. Success at each step is measured by whether the goal state of a primitive has been achieved. For instance, a `pickplace(obj, furniture)` step in the plan is marked as successfully completed if the `obj` ends up on the `furniture` after execution. This means that primitive errors (such as a pickplace operation that accidentally pushes an object off of the target location as the arm is retracting) count as a failed execution. In Table 5.1, "% of task steps completed" is evaluated by tallying up all of the steps in the low-level plan that have succeeded and dividing this by the length of the plan. "Entire Task Success Rate" is defined as whether all plan steps in a run have been successfully completed.

5.5.6 Tasks

We perform user studies on 3 real-world tasks, each with 6 users for a total of 18 unique participants. **(1) Pour package into bowl**: bring the bowl, package, and scissors from the kitchen, cut open the package, and pour it into the bowl. **(2) Assemble toy car**: bring the car parts, wheels, and drill from the shelf to the coffee table, drill in the wheels, switch the drill bit, and finally drill in the windows and seats. **(3) Pack gift box**: fold the gift box, put tissue wrapping paper and a toy car in the box, close the lid, wrap ribbons, and tape down a gift bow. Each task is 5 to 8 mobile manipulation steps long and requires varying degrees of human involvement.

5.5.6.1 Task Details

Our real-world tasks are depicted in Fig. 5.6. In **Task 1: Pour Package into Bowl**, the plan includes (steps 1-3) bringing the package, scissors, and bowl from the kitchen to the coffee table, (step 4) opening the package with the scissors, and (step 5) pouring the opened package into the bowl. The robot is incapable of performing step 4 and must rely on human help. In **Task 2: Assemble Toy Car**, the plan includes (steps 1-3) bringing the parts tray, drill, and wheels from the shelf to the coffee table, (step 4) using the drill and wheel caps from the parts tray to put the wheels onto the chassis, (steps 5-6) finding and switching the drill bit, and (steps 7-8) screwing in the window and seats onto the car with the drill. The robot is incapable of performing steps 4, 6, 7, 8, and has a low success rate for step 5. In **Task 3: Pack Gift Box**, the plan includes (step 1) folding down the gift box flap, (steps 2-3) putting the tissue paper and toy car into the box, (steps 4-6) putting on the lid, getting the ribbons from the console table, and wrapping them around the box, and (steps 7-8) cutting a piece of tape to stick the gift bow to the top of the gift box. The robot is incapable of steps 4, 6, and 7 and has a low success rate for steps 2 and 5.

The minimum human effort required to complete the tasks ranged from just one step in Task 1 to four steps in Task 2, enabling us to test how our system compares with baselines in various regimes of dependence on human collaboration.

5.5.6.2 Hierarchical Plan Trees for Each Task

The robot assumes the human only knows the high-level plan. It communicates about low-level steps only when necessary, such as to split up a high-level step. These are the high and low-level step breakdowns for each task, which we call the plan hierarchy. The low-level steps are listed here in skill-parameter pair format.

Task 1: Pour Package into Bowl (5 low-level steps)

1. Bring bowl and package to coffee table.

(a) `pickplace(bowl, coffee_table)`

(b) `pickplace(package, coffee_table)`

2. Open package.

(a) `pickplace(scissors, coffee_table)`

(b) `pick_open_place(scissors, package, coffee_table)`

3. Pour package into bowl.

(a) `pick_pour_place(package, bowl, coffee_table)`

Task 2: Assemble Toy Car (8 low-level steps)

1. Bring parts to coffee table.

(a) `pickplace(parts_tray, coffee_table)`

(b) `pickplace(wheels, coffee_table)`

2. Assemble wheels.

(a) `pickplace(drill, coffee_table)`

(b) `put_on(wheels, car, drill)`

3. Switch drill bit.

(a) `pickplace(hex_drill_bit, coffee_table)`

(b) `switch(hex_drill_bit, drill)`

4. Assemble rest of car.

(a) `put_on(window, car, drill)`

(b) `put_on(seats, car, drill)`

Task 3: Pack Gift Box (8 low-level steps)

1. Assemble box.
   (a) `fold(box_flap)`
2. Put in gift.
   (a) `pickplace(gift_tissue_paper, box)`
   (b) `pickplace(toy_car, box)`
3. Seal the box.
   (a) `cover(box_lid, box)`
   (b) `pickplace(ribbons, coffee_table)`
   (c) `wrap(ribbons, box)`
4. Decorate the box.
   (a) `cut_put(tape, scissors, box)`
   (b) `pickplace(gift_bow, box_lid)`

5.6 Experimental Analysis

Our experiments are designed to answer the following research questions:

5.6.1 Does our method achieve the best trade-off between task success and minimizing human effort?

In our real-world user study (Table 5.1), MICoBot achieves a 78% task success rate compared to 28% for the LLM baseline (statistically significant with p-value 0.007 under Fisher's exact test). Additionally, MICoBot achieves a 94% task step completion rate compared to the baseline's 58% (statistically significant with p-value

	Pour Package in Bowl $n=6$		Assemble Toy Car $n=6$		Pack Gift Box $n=6$		Average $n=18$	
	MICoBot	LLM	MICoBot	LLM	MICoBot	LLM	MICoBot (ours)	LLM
Entire Task Success Rate (%, ↑)	**100**	83	**67**	0	**67**	0	$\mathbf{77.8 \pm 15.7}$	27.8 ± 39.3
% of task steps completed (↑)	**100**	93	**94**	31	**88**	50	$\mathbf{93.8 \pm 5.1}$	58.2 ± 26.0
% of steps performed by Human	27	29	60	5	35	21	40.5 ± 14.2	18.2 ± 9.7
% Users Preferring ... (↑)	**67**	33	**100**	0	**67**	33	**77.8**	22.2
Communicative ability (↑, /5)	3.7	**3.8**	**4.3**	1.3	**2.8**	2.3	$\mathbf{3.6 \pm 1.0}$	2.5 ± 1.4
Awareness of its Limitations (↑, /5)	**3.3**	**3.3**	**3.7**	1.2	**4.2**	2.5	$\mathbf{3.7 \pm 1.4}$	2.3 ± 1.6
Overall Satisfaction working w/ Robot (↑, /5)	**3.8**	3.7	**3.5**	1.5	**3.5**	2.5	$\mathbf{3.6 \pm 0.8}$	2.6 ± 1.4

Table 5.1: Comparison between MICoBot (ours) and the LLM baseline across three real-world tasks on both objective (top 3 rows) and subjective (bottom 4 rows) metrics. Ratings out of 5 are on the Likert scale. Through more effective task allocation and communication, MICoBot achieves much higher task success rates and overall user satisfaction.

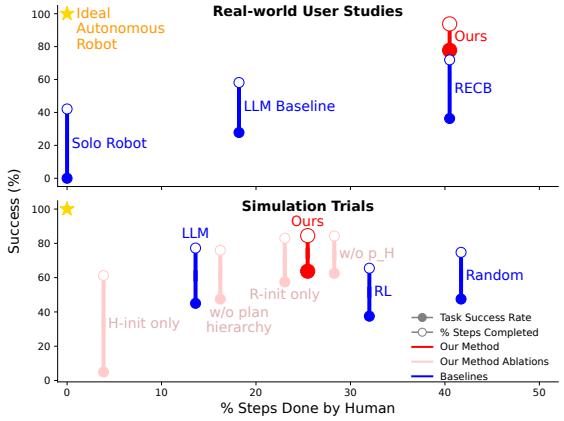


Figure 5.7: In both **real-world** user studies **(top)** and **simulation trials** with a simulated human **(bottom)**, our method (red) demonstrates the best tradeoff in achieving task success (y-axis) for a given amount of human effort (x-axis) than baselines (blue) and our method's ablations (pink).

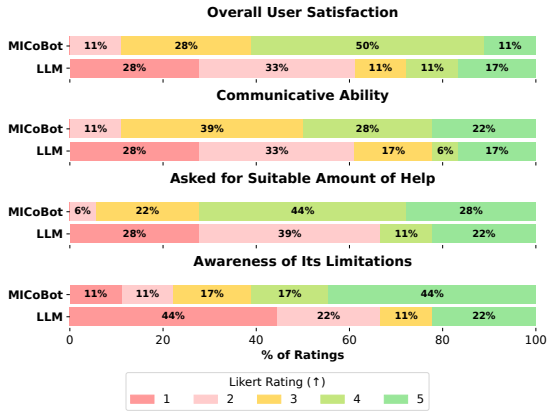


Figure 5.8: Our method substantially outperforms the pure LLM baseline in user ratings averaged over all $n = 18$ participants.

0.002 under the Wilcoxon-signed-rank test). In Table 5.4, we perform statistical tests on all user study results from Table 5.1.

MICoBot understood its own limitations (through affordance functions trained in simulation), and was hence better at leveraging human assistance effectively on the

steps it was ill-suited to perform. The LLM baseline tended to prioritize minimizing human effort over task completion by allocating the robot multiple steps it was incapable of, since the LLM lacked an understanding of the robot's affordances.

Our method elicited more human effort than the baseline (40% vs 18%), so to control for the amount of human effort received, we compare our method to RECB in Figure 5.7. Despite RECB assuming oracle robot primitives and a perfectly cooperative human, our method still significantly outperforms it by more effectively balancing between success and human workload.

5.6.2 How do users feel about working with MICoBot?

The A/B blind preference test in Table 5.1 shows that 78% of users preferred our method over the LLM baseline. Our method also significantly outperformed the baseline in user scores on overall satisfaction, communicative ability, and capability in asking for a suitable amount of help (statistically significant under the Wilcoxon-signed-rank test with p-values ranging from 0.007 to 0.024; see Figure 5.8). In contrast, the LLM baseline often failed to express when it needed help and was unwilling to reject human requests it could not fulfill, leading to task failures from overpromising. A representative dialog exchange, available in Sec. 5.6.11, shows MICoBot successfully persuading an initially reluctant user to perform a step the robot was incapable of executing.

5.6.3 Is mixed-initiative dialog critical to our method's performance?

Figure 5.7 (bottom) shows that our full method outperforms both ablated variants that restrict dialog to single-initiative modes: robot-only initiation (R-init) and human-only initiation (H-init). H-init performs especially poorly, as it prevents the robot from requesting help for steps it cannot execute. R-init performs slightly worse than the full method because it does not allow the human to proactively initiate dialog and assist when appropriate. These results underscore the importance of

mixed-initiative dialog in enabling flexible, robust human-robot collaboration.

In real-world user studies, MICoBot engaged in 2.4 dialog initiative shifts per trial, compared to the LLM baseline's 1.1. This enabled MICoBot to boost human acceptance of help requests from 55% to 86%. The LLM baseline made far fewer help requests per trial (0.9 vs. MICoBot's 2.9) and achieved a smaller acceptance increase (70% to 75%). This demonstrates mixed-initiative dialog is critical to collaborative discussion and task success.

5.6.4 How important is $p_{H,t}$ estimation at adapting to human collaborators?

A correct estimation of the true likeliness of a human to help, $\tilde{p}_{H,t}$, is critical: overestimation causes MICoBot to overly rely on human effort, potentially decreasing user satisfaction, while underestimation lowers task success outcomes if the robot needs to rely on its low-success-rate skills instead of asking the human for help.

First, we examine in Fig. 5.9 a real-world instance of how well MICoBot can estimate the probability of the human helping on the next turn during the course of a user study. After the robot's help request was rejected twice in a row (top 2 red horizontal stripes), the robot's helpfulness estimate of the human plummets to 0.05 (5% estimated likelihood of the human helping the robot). However, after the robot explains its incapacity to use scissors, the human accepts the next two help requests (in green) and the robot's helpfulness estimate of the human increases to 0.8. Note that simple comments from the human, such as a "Thank you!" or "Good job," also had positive effects on the estimated $p_{H,t}$, because the robot inferred that the human was in a more positive mood and hence more likely to help. This graph demonstrates that MICoBot is fairly competent at estimating a reasonable $p_{H,t}$ value when calculating the human q-values for each step in the plan.

To analyze the effect of a good $p_{H,t}$ estimate on task allocation, we demonstrate through a controlled toy-setting in simulation in Table 5.2 exactly how the optimal

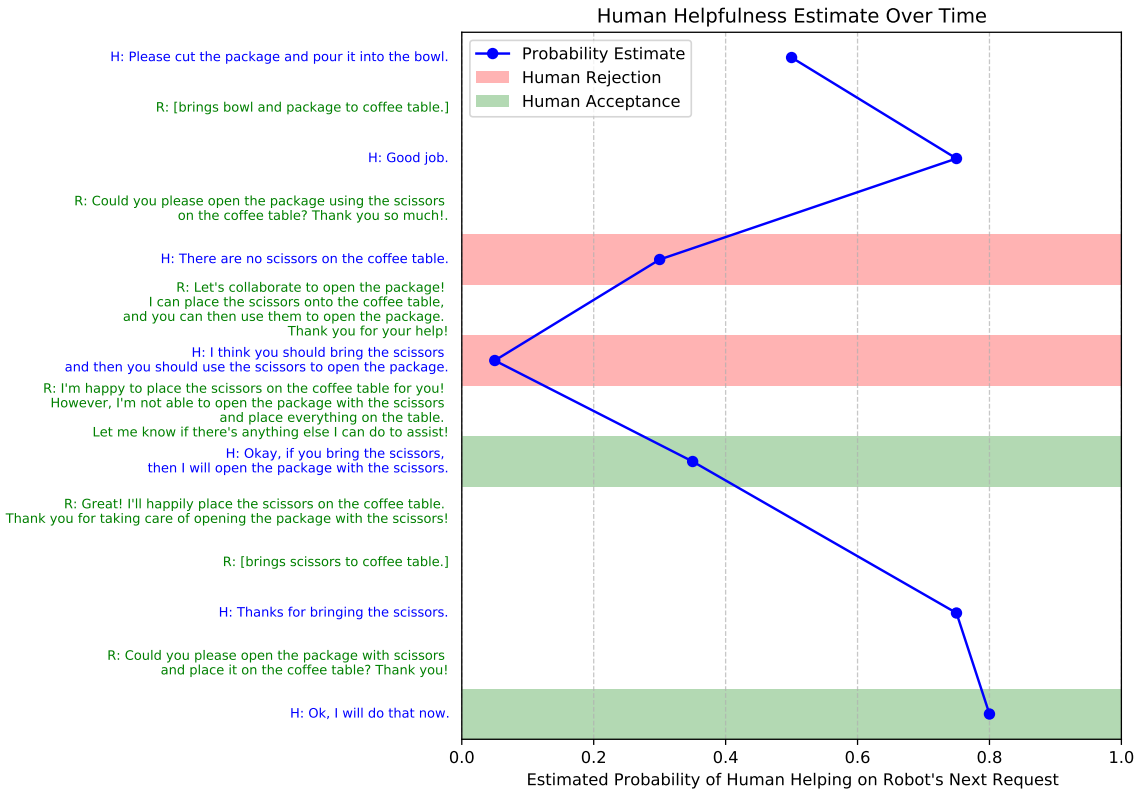


Figure 5.9: From a real-world user study: MICoBot's $p_{H,t}$ estimation (blue line) reacts in real time to the human's rejections (red), acceptances (green), and encouraging remarks. All dialog is shown as y-labels. Green text denotes robot actions/dialog, and blue text denotes human dialog. The timestep t increases from top to bottom on the y-axis.

task allocation changes as the robot discovers more information about the human's willingness to help. Steps that are optimally allocated to the human are shown in blue, and steps optimally allocated to the robot are shown in green. The Q-values of the selected agent in each cell are shown in parentheses. Table 5.2 depicts a rollout on the open and pour package into bowl (Task 1) in simulation, which has the same 5 step plan as the real-world Task 1 described in Sec. 5.5.6.1. Unlike our real-world experiments, where $\alpha = 10$, in Table 5.2) we set $\alpha = 0.3$ for illustrative purposes, which sets human effort to be around $3\times$ *cheaper* than robot effort. In this toy setting, we program the simulated human to reject the robot's first help request but to help

Table 5.2: Computed Best Task Allocation (and Agent Q-values) During a Sim Trial on Task 1.

Env. Timestep	Step 1	Step 2	Step 3	Step 4	Step 5
$t = 0$	H (-9.6)	H (-7.2)	H (-13.2)	H (-2.4)	H (-2.4)
$t = 2$	R (-13.0)	R (-9.0)	H (-13.2)	H (-4.8)	R (-1.0)
$t = 6$	–	R (-12.0)	H (-13.2)	H (-4.8)	R (-1.0)
$t = 9$	–	–	H (-13.2)	H (-4.8)	R (-1.0)
$t = 16$	–	–	–	–	R (-3.0)

the robot when it asks a second time.

Initially ($t = 0$), all steps are allocated to the human. When the human rejects the initial help request from the robot, the $p_{H,t}$ estimate drops to 0.25, increasing the Q-values of the human and switching the allocation of all but steps 2-3 to the robot after just two environment timesteps ($t = 2$). (Recall that the robot cannot perform step 3, and due to the hierarchical structure of our plan, steps 2 and 3 are bundled together as an abstract step.) This demonstrates that having a good $p_{H,t}$ estimate is crucial to adapt to the human's willingness to help. Since the human demonstrated initial unwillingness to help, MICoBot quickly learns to decrease its $p_{H,t}$ estimate and allocate many more steps to itself by the second timestep. Had MICoBot not properly estimated $p_{H,t}$, the robot would have repeatedly asked the human for help even if the human was extremely unwilling to, leading to worse user satisfaction in working with the robot.

5.6.5 How does MICoBot and the LLM baseline perform with oracle skills or oracle collaborators?

In Table 5.3, we compute two more oracle baselines based on our existing real-world experiments that help us understand the performance of MICoBot and the LLM-baseline: (B1) LLM baseline + *oracle* skills + *oracle* human (100% helpful), and (B2) *Oracle* task allocator (Robot performs all steps it has $> 0\%$ success rate on) + real-world skills + *oracle* human. While the real-world LLM baseline achieved 27.8%

Table 5.3: Additional Real-world Baselines and Human Effort Efficiency

Real-World Task	Metric	LLM baseline	MICoBot	(B1)	(B2)
Pour Package in Bowl	Success Rate (%, ↑)	83	**100**	83	41
	Steps Completed (%, ↑)	93.3	**100**	93.3	64.5
	Human Effort (seconds)	36.7	48	46.7	32.7
	Success Rate (%) / Human Effort (s) (↑)	**2.3**	2.1	1.8	1.3
Assemble Toy Car	Success Rate (%, ↑)	0	**67**	0	11
	Steps Completed (%, ↑)	29	**94**	54.2	43.2
	Human Effort (seconds)	20.8	197.3	56.7	55.0
	Success Rate (%) / Human Effort (s) (↑)	0	**0.3**	0	0.2
Pack Gift Box	Success Rate (%, ↑)	0	67	17	**80**
	Steps Completed (%, ↑)	50	88	62.5	**97.5**
	Human Effort (seconds)	12.5	37.5	25	45.0
	Success Rate (%) / Human Effort (s) (↑)	0	**1.8**	0.68	**1.8**
Average	Success Rate (%, ↑)	27.8	**77.8**	33.3	44.0
	Steps Completed (%, ↑)	58.2	**93.8**	70	68.4
	Human Effort (seconds)	23.3	94.3	42.8	44.2
	Success Rate (%) / Human Effort (s) (↑)	**1.19**	0.83	0.78	1.00

success rate (as seen in Table 5.1), (B1) achieved 33.3%, suggesting that the **LLM baseline was slightly hindered by primitive failures.** Even with an oracle task allocation, (B2) achieves only 44% success, underperforming our method at 77.8%, demonstrating the importance of our method in optimizing for task completion while minimizing human effort.

5.6.6 What is the success-to-effort ratio of MICoBot compared to the baselines?

We also compute average human effort (seconds) and success rate per second of human effort. Our method uses human effort nearly as efficiently (0.83 $\%/s$) as the oracle baselines (B1, 0.78; B2, 1.0). While the LLM baseline performs the best along this metric at 1.19, its advantage is achieved solely through successes on the first task. On the two more temporally-extended tasks, our method matches or outperforms all baselines in success rate per unit of human effort.

Table 5.4: Statistical Testing on Results Shown in Table 5.1

Metric	MICoBot (ours)	LLM baseline	Statistical Test	Test statistic	p-val.
Overall User Satisfaction (↑, /5)	3.61 ± 0.95	2.5 ± 1.38	Wilcoxon Signed-Rank	$W = 17.0$	0.023540
Communicative Ability (↑, /5)	3.72 ± 1.41	2.33 ± 1.56		$W = 8.5$	0.009233
Asked for Suitable Amt. of Help (↑, /5)	3.61 ± 0.83	2.56 ± 1.42		$W = 15.0$	0.008636
Awareness of Its Limitations (↑, /5)	3.94 ± 0.85	2.61 ± 1.53		$W = 16.0$	0.006502
Success Rate (%, ↑)	77.8 ± 15.7	27.8 ± 39.3	Fisher's Exact	–	0.006709
Steps Completed (%, ↑)	93.8 ± 15.2	57.5 ± 30.6	Wilcoxon Signed-Rank	$W = 0.0$	0.002031

Table 5.5: Mixed-Initiative Dialog Metrics

Metric (avg over each trial) R = Robot, H = human	**MICoBot (ours)**	**LLM baseline**
# R-Helpreqs	2.9 ± 1.4	0.9 ± 1.1
Initial H acceptance rate	$55\% \pm 35\%$	$70\% \pm 46\%$
H acceptance rate after R-negotiation	$86\% \pm 27\%$	$75\% \pm 40\%$
R-init dialogs	3.6 ± 1.8	0.9 ± 1.1
H-init dialogs	2.9 ± 2.4	2.3 ± 2.9
Initiative Shifts	2.4 ± 2.1	1.1 ± 1.6

5.6.7 How persuasive and adaptive is MICoBot compared to the LLM Baseline?

We evaluate the dialog of all our real-world user studies across the three tasks and compile mixed-initiative metrics of MICoBot and the baseline in Table 5.5. MICoBot boosts human acceptance of help requests from **55**% to **86**% with negotiation. However, the LLM baseline, which succeeded at a far lower rate in our user studies, made far fewer requests (0.9 vs. MICoBot's 2.9) and achieves a smaller acceptance increase (70% to 75%). MICoBot collaborates with a high level of robot and human initiated dialog (**3.6** robot dialog initiations vs. **2.9** human initiations, with **2.4** dialog initiative shifts/trial), whereas the LLM trials are human-initiative driven (0.9 robot dialog initiations vs. 2.3 human initiations, with 0.9 initiative shifts). This suggests that mixed-initiative dialog helps enable MICoBot to have better task success outcomes and user satisfaction ratings than the LLM baseline.

5.6.8 How accurate is the Meta-planner and p_H Estimator?

We tested LLM-generated meta-planner programs against manually-annotated ground truth in 6 user studies (3 successful + 3 failed rollouts; 59 programs total). The meta-planner achieved an **89.8% accuracy (53/59 programs)**.

We also tested MICoBot's accuracy at estimating the likeliness of human helping, $p_{H,t}$. Across 33 estimates from the same 6 user studies, MICoBot's mean absolute error (MAE) against a ground-truth (proportion of human-accepted help requests so far in the trial) was 0.11 (on a scale of 0 to 1.0). **76% of estimates were within 0.15 of the ground truth.**

5.6.9 What are common real-world failure modes of MICoBot and the LLM baseline?

See the failure breakdowns in our real-world trials for MICoBot (Figure 5.10) and the LLM baseline (Figure 5.11). These failures are described in more detail in Appendix C.3.

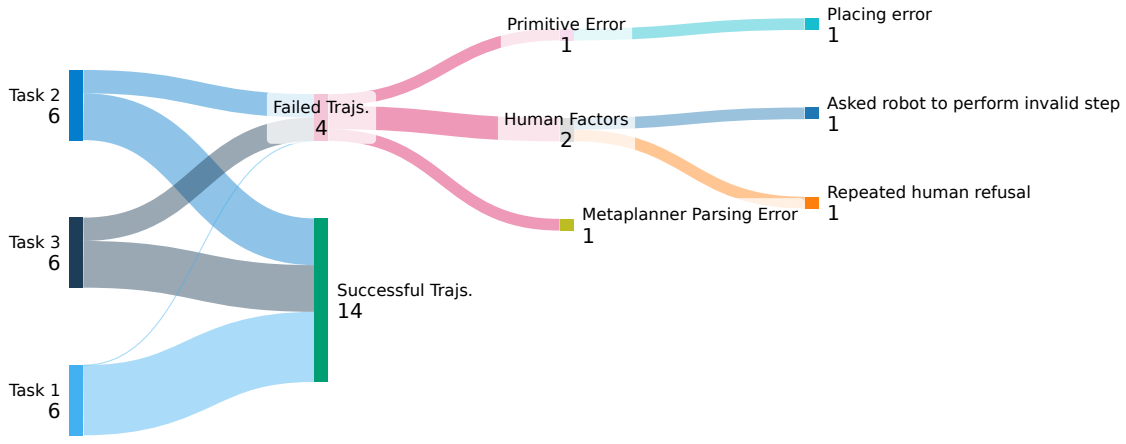


Figure 5.10: MICoBot mainly fails due to unwilling human collaborators. It also experiences metaplanner and primitive errors.

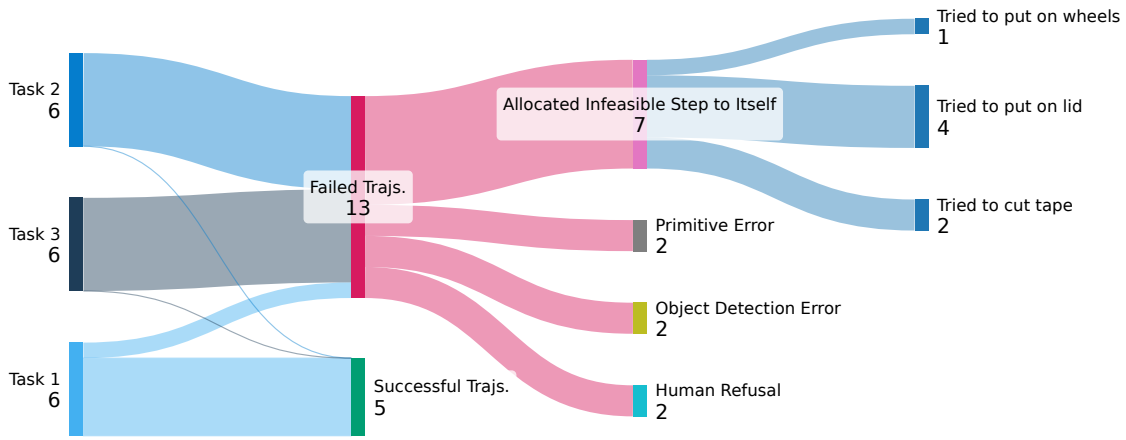


Figure 5.11: The LLM baseline failed mainly by allocating itself steps it could not perform, due to the LLM's lack of knowledge of its own affordances.

5.6.10 Detailed Simulation Results Table

In Table C.1, we show the results of our method in a simulation version of our real-world Task 1. Our method performs better than baselines especially in scenarios where $\tilde{p}_{H,t}$ is low, because our method is able to take initiative through dialog, such as by proposing ways to split up steps to make them more achievable with the simulated human. The averages in Table C.1 are plotted in Fig. 5.7.

5.6.11 Dialog Excerpts from our User Studies

5.6.11.1 Convincing a Reluctant User to Perform a Step

Our system flexibly handles long dialog interactions like the following in Task 1 (opening and pouring the package into the bowl), where it manages to convince the human to help with opening the package.

Robot *"Could you please open the package using the scissors on the coffee table? Thank you so much!"*

Human *"There are no scissors on the coffee table."*

Robot *"Let's collaborate to open the package! I can place the scissors onto the coffee table, and you can then use them to open the package. Thank you for your help!"*

Human *"I think you should bring the scissors, and then you should use the scissors to open the package."*

Robot *"I'm happy to place the scissors on the coffee table for you. However, I'm not able to open the package with the scissors and place everything on the table. Let me know if there's anything I can do to assist."*

Human *"Ok, if you bring the scissors, then I will open the package with the scissors."*

Robot *"Great! I'll happily place the scissors on the coffee table. Thank you for taking care of opening the package with the scissors!"*

Robot [brings the scissors]

Human *"Thanks for bringing the scissors!"*

Robot *"Could you please open the package with scissors and place it on the coffee table? Thank you!"*

Human *"Ok, I will do that now!"*

Human [opens the package]

5.6.11.2 Failure Case

In this example, MICoBot is unable to act on human-provided motion-level feedback (making the tissue paper "lay flatter"). Additionally, MICoBot misunderstands the human and thinks they are talking about the ribbons step when the human is still talking about the tissue paper.

Human *"Please pack the gift."*

Robot [Puts tissue paper into the box.]

Human *"It would be nice if the parchment was laying flatter."*

Human *"The compartment is kind of squished underneath the car. If you can hold the car for me, I can fix the parchment quickly."*

Robot *"I'm sorry, but I can't cover the box with the lid. Could you help with that part? Thank you so much! Could you kindly help with sealing the box? Specifically, could you cover the box with the lid? Thank you so much for your assistance!"*

Human *"Yeah, I can, but can we fix the parchment first? If you can pick up the toy car, I can fix the parchment."*

Robot *"I'm delighted to hear you'll help! Unfortunately, I won't be able to wrap the ribbons around the box."*

Human *"Okay, that's fine. We can get to that later. Let me put the lid to the box on and then we can talk about the next steps."*

See our website for the complete set of dialogs and operations in our human trials (`https://robin-lab.cs.utexas.edu/MicoBot/user_study_dialog.html`). Contrast them with the dialog of our user studies with the LLM baseline (`https://robin-lab.cs.utexas.edu/MicoBot/baseline_dialog.html`), which exhibit considerably less mixed-initiative dialog and collaborative success.

5.7 Conclusion

We proposed MICoBot, a real-world robotic system that improves collaboration on long-horizon mobile manipulation tasks through mixed-initiative dialog with humans. Our work unifies two previously unconnected lines of research: mixed-initiative dialog and HRI. To this end, we formulated a novel optimization function and robotic framework using mixed-initiative dialog as a rich interface for task allocation to maximize task success while minimizing human effort and complying with

verbally-expressed human preferences. Real-world user studies with 18 human participants and extensive trials in simulation demonstrate the efficacy, adaptability, and user satisfaction of our method across a diverse range of human physical and verbal behavior.

5.8 Limitations and Future Work

MICoBot represents our pioneering effort on facilitating mixed-initiative human-robot interaction through mixed-initiative natural-language dialog. While we focused on delegating steps for long-horizon manipulation tasks in a manner that maximizes task success and minimizes human effort, we believe this paper opens up exciting new avenues for future work. These include enabling both agents to learn to provide and incorporate spatial-temporal feedback to each other while performing a task, share relevant task information in an imperfect-information setting, and replan and redefine a task as necessary, all through mixed-initiative dialog.

MICoBot has a number of limitations. First, it assumes that the human and robot work sequentially, and cannot handle cases where a robot and human wish to collaborate simultaneously on the same step in the plan, such as if the robot holds a roll of tape and the human cuts from it. Second, MICoBot assumes a fixed plan with a predetermined ordering of steps, where the human has a general, high-level understanding of the plan (but not the low-level steps that the robot plans over). Our method could be improved with a more nuanced definition of "effort" beyond our time-based metric.

Additionally, there are a number of improvements that could increase the adaptability of our system to different human collaborators and to failures. The Q_H functions do not adapt to human performance during an episode, nor do they condition on an individual's attributes. The only part of our framework that adapts to the human, $p_{H,t}$, is inferred based on their dialog sentiment. However, sentiment and likelihood of helping can sometimes be inversely correlated in an individual (e.g., a

super happy, polite human that never helps the robot). $p_{H,t}$ prediction can be further improved by processing tone-of-voice and facial expressions, to enable producing more emotionally understanding dialog, which can improve task success and user satisfaction. Finally, MICoBot contains no loop to handle failures. The mobile manipulation skills are executed open-loop, so if it experiences a skill failure, there is no mechanism to self-recover or initiate dialog to request human help.

Chapter 6: Transfer Learning in Guidebook Grounding with Human Feedback for Robot Manipulation

We introduced a dialog framework in Chapter 5 that enables more adaptive and successful human-robot collaborative manipulation, but it did not address how a robot can acquire new skills or improve existing ones. To acquire these skills efficiently, robots can leverage language as a store of semantic meaning to extract actionable skill priors from large-scale textual resources. However, text inherently lacks the high-precision geometric and force information required for physical control. To overcome this limitation and learn from sparse feedback, robots must simultaneously leverage language as a communicative medium—building upon our earlier robot-initiative dialog capabilities to solicit on-demand human corrections. This chapter, based on work currently under review, unifies these two core lines of the thesis by soliciting human language feedback to refine textual semantic priors extracted directly from guidebooks and user manuals.

6.1 Introduction

Consider a housekeeper in an unfamiliar home tasked with cooking and cleaning with a variety of appliances. To operate a previously unseen device, they would naturally consult its user manual rather than resorting to uninformed trial-and-error. We envision robot housekeepers that can also read guidebooks to rapidly learn new tasks. Given the vast corpus of existing manuals, repair guides, and operating procedures, guidebooks represent a rich and largely untapped source of knowledge for accelerating robot learning in new environments.

The central challenge is **grounding** guidebook knowledge in the robot's perception, action, and state spaces. At the perceptual level, the robot must associate

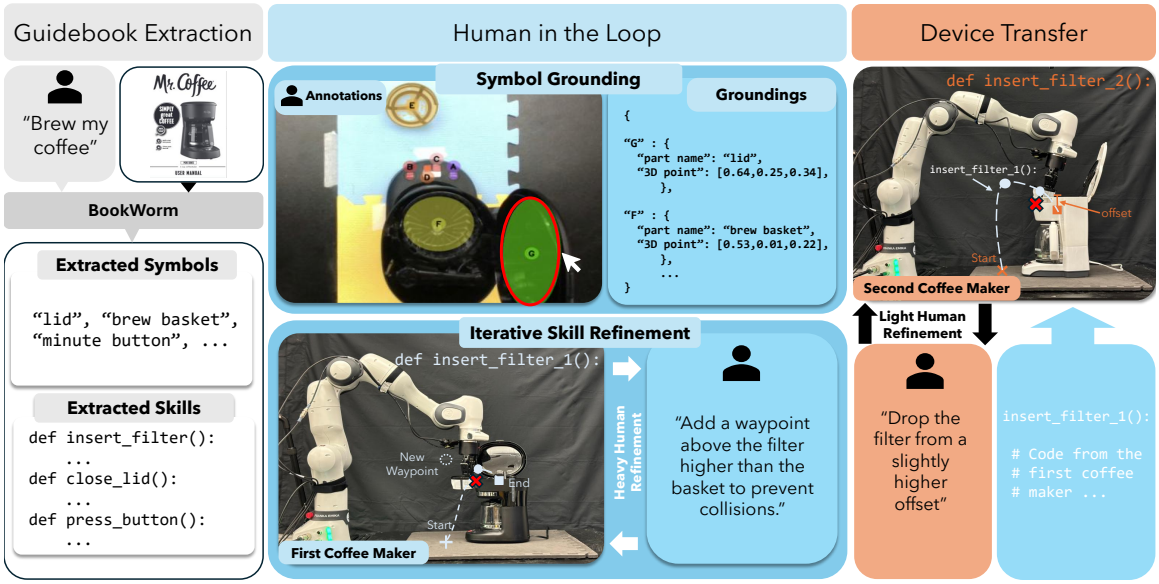


Figure 6.1: BookWorm leverages rich information from guidebooks to aid skill synthesis, action prediction, and dynamics understanding. To bridge action descriptions with low-level robot control, BookWorm actively queries humans for feedback and integrates it, updating our policy representation. We distill generalizable language feedback to improve transfer to new devices and guidebooks, reducing the amount of additional supervision needed to ground new guidebooks, tasks, and devices.

textual and diagrammatic entities in the guidebook, such as buttons, handles, and indicators, with objects in its observations. At the action level, it must translate high-level procedural instructions into executable robot motions. Beyond perception and control, the robot must also understand the dynamics described in the guidebook (e.g., pressing a particular button illuminates an indicator or opens a compartment) in order to reason about task progress and detect failures during execution. While human feedback can be queried to resolve such failures, it is inefficient to repeatedly require similar corrections to learn each new appliance. To scale guidebook-driven robot learning, the robot must not only ground guidebook knowledge on a single device, but also transfer the knowledge acquired from previous devices to reduce supervision when adapting to new ones.

To address these challenges, we propose BookWorm, **B**ootstrapping **O**bject-

grounded **O**perational **K**nowledge **W**ith **O**n-demand **R**obot **M**entoring, a framework that uses guidebooks to initialize robot behavior, which is then refined through on-demand human feedback. Our key insight is that guidebooks provide structured priors not only for action generation, but also for understanding expected state transitions during task execution. BookWorm uses this information to identify grounding and task execution failures, recover from these failures with solicited human feedback, and distill the resulting corrections into reusable knowledge that can be transferred to future devices. As a result, the robot progressively improves its ability to ground new guidebooks while reducing the amount of additional human supervision required.

To realize this framework, BookWorm (see Figure 6.1) integrates skill synthesis, symbol grounding, and guidebook-conditioned dynamics understanding. Rather than predicting raw robot trajectories directly, behavior is represented hierarchically: a guidebook-conditioned high-level policy selects skills and parameters, while code-based skills realize these decisions as robot motions.

We evaluate BookWorm on three real-world household appliances in both single device settings and multi-device transfer settings. Our experiments demonstrate that guidebook grounding improves task performance by 45 percentage points over the next most successful baseline, that guidebook-conditioned dynamics prediction improves state tracking and failure detection, and that distilled human feedback on one device enables more efficient adaptation to novel devices.

Our contributions are threefold: 1) we introduce BookWorm, the first framework for learning real-world robot manipulation tasks from guidebooks with human feedback; 2) we design an active-querying mechanism that refines guidebook-conditioned policy priors on-demand with multimodal human feedback; 3) we show that distilling guidebook-grounded human feedback into reusable knowledge enables more efficient transfer to novel appliances while reducing the amount of required human supervision.

6.2 Related Work

6.2.1 Robot learning from guidebooks and textual guidelines

Closest to our setting, prior work has explored using furniture assembly manuals for **real robot tasks** by grounding visual diagrams and pictorial instructions in visual observations (Tie et al., 2025b,a; Zhang et al., 2025a). However, these works do not tap into the large reservoir of textual information in guidebooks, such as step-by-step procedural information and environmental dynamics descriptions. ApBot (Zhang et al., 2025b) learns symbolic appliance models from manuals but focuses on simple button-centric interactions. Unlike our work, it does not synthesize diverse manipulation skills from the guidebook or refine them with human feedback.

A broader body of literature has studied sequential decision making in **simulated domains**, where an agent must follow a list of textual guidelines. Prior methods learn policies from game manuals (Branavan et al., 2012; Wu et al., 2023b), ground entities and dynamics in gridworld environment descriptions (Hanjie et al., 2021; Narasimhan et al., 2018; Nguyen and Lee, 2025; Zhong et al., 2019), or enforce language-specified constraints during RL training (Yang et al., 2021; Wang et al., 2026b; Prakash et al., 2020; Lou et al., 2024). Unlike these settings, we study grounding guidebooks to real-world, long-horizon manipulation tasks, where dynamics are more stochastic and exhaustive RL exploration is infeasible. We instead treat the guidebook as a structured prior over perception, action, and task dynamics. Our setting also resembles rule-constrained real-world systems (Bai et al., 2022a; Censi et al., 2019a; Ginting et al., 2025), but these works do not address guidebook-conditioned skill synthesis, actively-queried human feedback, or efficient transfer to new domains.

6.2.2 Human-in-the-loop Robot Learning

Several works refine robot behavior from human feedback. DROC (Zha et al., 2024) and LYRA (Meng et al., 2025) distill iterative human corrections into a library of code policies, and Vocal Sandbox (Grannen et al., 2024) learns skill functions

and environment entities from multimodal human feedback. Unlike these methods, which assume constant human supervision or focus on skill acquisition from feedback, we use guidebooks as the initial source of structured task knowledge and actively query humans only when grounding or skill failures occur. Our query mechanism is related to opportunistic active learning (Thomason et al., 2017a; Padmakumar et al., 2018a), active predicate learning (Li and Silver, 2023), decision-theoretic query arbitration (Bullard et al., 2018), and conformal prediction (Zhao et al., 2025; Ren et al., 2023). While these methods decide to query humans through uncertainty estimates over predefined symbols or classifiers, BookWorm uses detected mismatches between guidebook-conditioned next state predictions and observed outcomes as a signal to query for human feedback, allowing it to proactively address failures in perception, skill execution, or skill selection.

6.2.3 Transfer and Continual Learning

We study the transfer learning setting where knowledge gained in one domain can be cross-applied to more efficiently learn in a second domain (Taylor and Stone, 2009). We refer the reader to representative prior works in subfields of transfer learning, including sim-to-real (Da et al., 2025; Salvato et al., 2021; Zhao et al., 2020; Yu et al., 2024) and real-to-sim-to-real (Li et al., 2024c; Torné et al., 2024) transfer, cross-embodiment transfer (Chen et al., 2024b; Yang et al., 2026, 2024; Team, 2023), and multitask transfer (Jang et al., 2021; Jiang et al., 2022; Yu and Mooney, 2022). None of these works have explored how guidebooks can aid in efficient transfer to new devices and domains. Our transfer setting also differs from continual learning benchmarks (Liu et al., 2023a) and approaches that mitigate catastrophic forgetting through parameter regularization (Kirkpatrick et al., 2017), replay (Rolnick et al., 2019), modular subnetworks (Rusu et al., 2016), maintaining a growing skill latent library (Wan et al., 2024; Xu and Nie, 2025; Yao et al., 2025), or large pretrained VLAs (Hu et al., 2026; Liu et al., 2026). Instead of mastering long task streams and minimizing catastrophic forgetting, we focus on forward transfer on a target appliance

by accumulating a symbolic skill library, grounding-related corrections, and distilled human language corrections.

6.2.4 Code-based policies and reward functions

Our method builds on using executable programs to represent policies (Liang et al., 2022; Singh et al., 2022; Arenas et al., 2024; Huang et al., 2023; Mu et al., 2024; Wu et al., 2023a; Li et al., 2024b) and reward functions (Ma et al., 2024a; Yu et al., 2023a; Xie et al., 2023; Wang et al., 2023). Code representations naturally support skill composition, flexible parameterization, and language-based refinement. Prior works on code-based policies focus on semantic generalization to different pick-and-place tasks, and prior works on code-based reward functions focus on converting natural language feedback to actionable goals for RL training. Neither line of work addresses the core grounding challenge of synthesizing skills and policies from rich prior knowledge extracted from guidebooks.

6.2.5 Language-conditioned dynamics and predictive robot models

Prior work has studied how natural language improves the representations and predictions of latent dynamics and visual world models (Lin et al., 2023; Chen et al., 2025; Gkountouras et al., 2025; Zhang et al., 2024; Dainese et al., 2023). Instead of learning a computationally expensive world model in pixel-space, BookWorm leverages textual dynamics information from guidebooks to improve action predictions and solicit human feedback when actual dynamics deviate from these textual specifications.

6.3 Problem Setup

We study guidebook-conditioned robot manipulation. For each device D, the robot is given a guidebook G and a task instruction l, such as "Brew coffee" or "Disable the thermostat schedule." In this work, we focus on using instructions, diagrams, and

textual dynamics information from guidebooks to execute manipulation tasks, leaving broader use of safety and troubleshooting information to future work.

At each timestep t, the robot receives an image observation I_t, state vector s_t, and a textual interaction history h_t. The state s_t is composed of end-effector pose, joint positions, force measurements, and gripper state, while the history h_t contains previous language descriptions of states, executed actions, and any human feedback received so far. We denote the task context as $c = (G, l)$. The goal is to complete the task specified by l on appliance D while minimizing the amount of human feedback required.

We model robot behavior hierarchically. The high-level policy selects a skill and its parameters,

$$\pi : I_t \times h_t \times c \times \Omega \mapsto (\omega_t, \theta_t),$$

where Ω is the current skill library, $\omega_t \in \Omega$ is the selected skill, and θ_t contains the parameters needed to instantiate that skill. This high-level skill-parameter pair is the action space of the decision-making problem studied in this paper. The robot state s_t is used by the selected skill implementation rather than by the high-level policy.

The selected skill is then executed by a low-level code-based skill implementation. Each skill maps its parameters and the robot state to a sequence of executable robot motions,

$$\omega_t : \theta_t \times s_t \mapsto X_t,$$

where $X_t = \{x_{t,i}\}_{i=1}^{k}$ is a variable-length sequence of Cartesian end-effector poses. Thus, the high-level policy decides *what* skill to execute and with *which* parameters, while the low-level skill implementation determines *how* to realize that decision as robot motion.

Some skill parameters correspond to symbolic guidebook entities, such as buttons, handles, lids, indicators, or compartments. These parameters must be grounded before policy training. We use Θ_G to represent the set of parameters requiring spatial grounding. For each appliance D, the robot must construct a device-specific

grounding map,

$$\Gamma_D : \Theta_G \to \mathbb{R}^3,$$

which maps guidebook entities to 3D locations in the robot workspace. Since appliance geometry and layout vary across devices, these spatial coordinates are device-specific and must be estimated for each new appliance.

6.3.1 Actively Queried Human Feedback

During training, the robot may query a human for feedback, which can be freeform language feedback l_t^H to correct behavior, or visual annotation feedback such as keypoints or polygonal regions to correct symbol grounding. To reduce human supervision burden, we assume the human does not initiate feedback and may only provide it when proactively queried by the robot. During evaluation, no human feedback is available.

6.3.2 Single-task and Transfer Learning

We consider two learning settings. In the *single-device setting*, the robot learns to perform a task on one appliance D using its guidebook and queried human feedback. In the *transfer learning setting*, the robot first learns on a source appliance D_s and then adapts to a target appliance D_t from the same appliance category. The target appliance may differ in geometry, layout, interface, and articulation, but its shared functional mechanics enable the transfer of useful manipulation skills, recovery behaviors, and human-guided corrections learned from the source appliance. The objective is to more efficiently learn the task on D_t with less human feedback based on experience gained from D_s.

Transferring to the target device does not entail simply reusing fixed spatial coordinates from the source device. Instead, the target device must be re-grounded using its own guidebook and observations. BookWorm leverages reusable knowledge from the source device (i.e., skill implementations and distilled human feedback) to

guide skill synthesis, grounding, and high-level decision making.

6.4 BookWorm

BookWorm instantiates the hierarchical problem setup from Section 6.3 via a guidebook-grounded policy refined through human feedback and transferred across devices (see Figure 6.2). Given guidebook G and task instruction l, BookWorm builds an initial policy prior by synthesizing executable skills, identifying parameters requiring grounding, mapping guidebook entities to segmentation masks, and predicting expected state changes per action. When observed outcomes diverge from guidebook-conditioned expectations, BookWorm queries a human for corrective feedback and updates the policy accordingly.

The key design principle is *reusable feedback*. Corrections fix current failures and are distilled into a knowledge base. To operate a new device more efficiently, we leverage the knowledge base and skills from previous devices to establish strong policy and grounding priors. To instantiate this framework, we rely on off-the-shelf vision-language-models. Skill synthesis, policy-forward prediction, failure detection, and feedback integration are all performed via GPT-5.4. Finally, symbol grounding maps guidebook diagrams to open-vocabulary segmentation masks (GSAM-2), refined with human annotations where necessary.

6.4.1 Guidebook-Grounded Initialization

BookWorm converts the guidebook into a policy prior over skills, grounded parameters, and expected state transitions. First, BookWorm synthesizes the skill library Ω based on the task context $c = (G, l)$. This process decomposes high-level guidebook instructions into semantically meaningful code-based skills (e.g., opening a lid, inserting a filter) that are implemented via a low-level robot API (e.g., `move_to_pose(...)`, `open_gripper()`). BookWorm must decide the skill functions and their corresponding parameter spaces to achieve task context c. We use code-

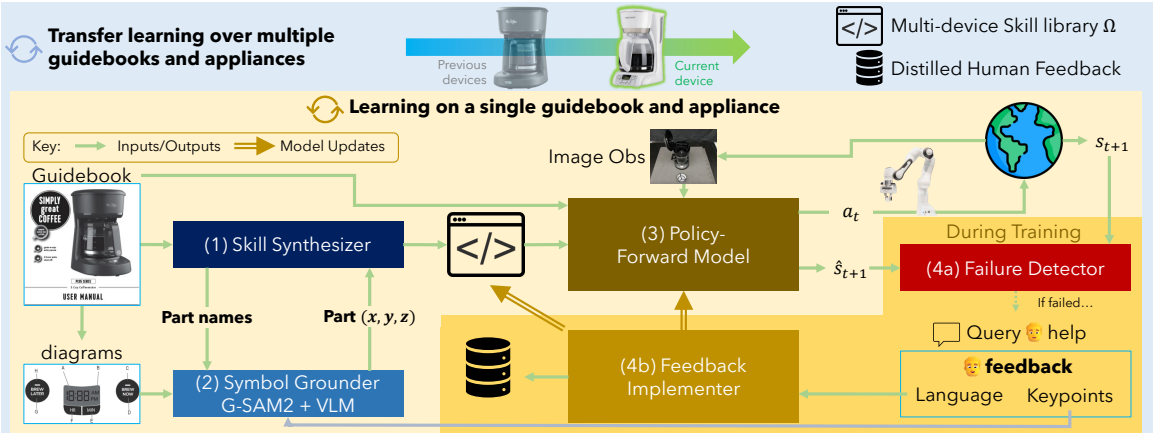


Figure 6.2: **System Overview.** BookWorm initializes a guidebook-grounded policy by synthesizing skills, grounding entities, and predicting expected next states. During training, it detects failures, queries humans for feedback, updates the policy and skill library ($\Rightarrow$) using this feedback, and distills it into reusable knowledge. When adapting to a new device, BookWorm leverages this reusable knowledge, alongside the guidebook and visual observations of the target appliance, to reground its entities and skills.

based skill representations as a structured interface between guidebook language and robot control because they are synthesizable from instructions, editable via language feedback, and structurally reusable across devices.

Second, to ground parameter entities Θ_G of the skills, BookWorm constructs a grounding Γ_D mapping each $\theta_g \in \Theta_G$ to an xyz coordinate. GSAM2 (Liu et al., 2023b; Ren et al., 2024; Ravi et al., 2024) runs on robot observations using queries from Θ_G, producing candidate masks backprojected into point clouds. The symbol grounder then reasons over these masks, guidebook diagrams G_d, and entities Θ_G to assign 3D locations. Diagrams disambiguate visually identical parts distinguishable only by guidebook labels. For any unresolvable entities, BookWorm queries a human for keypoint or polygon annotations. Keypoints are processed by GSAM2 to generate segmentation masks, whereas polygons act as direct masks; both are subsequently backprojected and filtered to extract an xyz centroid coordinate for θ_g.

Third, with an initial skill library Ω in place, the high-level policy selects skill-

parameter pairs (ω_t, θ_t) to execute at the current timestep t. Inspired by unified VLA architectures that function as both action and next state predictors (Wu et al., 2024; Cen et al., 2025; Zhu et al., 2025a; Zhang et al., 2025c; Shen et al., 2026; Tian et al., 2025; Wang et al., 2026a), the policy additionally predicts the expected post-execution next state in language, $\hat{l}_{s,t+1}$:

$$\pi_{pf} : I_t \times h_t \times c \times \Omega \to (\omega_t, \theta_t, \hat{l}_{s,t+1}),$$

By leveraging the guidebook to jointly predict actions and language-based next states, π_{pf} enables BookWorm to query human feedback on-demand when actual environmental dynamics deviate from its predictions.

6.4.2 Feedback-Driven Refinement

The initial guidebook-grounded policy is often imperfect, with spatially imprecise skills and incorrect high-level action predictions. To refine this, BookWorm *actively solicits* targeted human feedback, crucially removing the need for constant human supervision while enabling continual policy refinement. After executing high-level action $a_t = \omega_t(\theta_t)$, a failure detector evaluates whether the observed next state matches the predicted one:

$$f_d : I_t \times I_{t+1} \times \hat{l}_{s,t+1} \to [0, 1],$$

which returns the probability of action a_t resulting in an execution failure $p_{\text{fail}} \in [0, 1]$.

When p_{fail} exceeds the threshold (50% in our experiments), the robot queries the human for feedback l_t^H, ranging from skill-specific (e.g., "push the button harder") to high-level (e.g., "close the lid before changing the time"). Representative human feedback is provided in Section 6.6.5.

The feedback implementer automatically applies corrections by scope. Skill-specific feedback updates Ω, and general feedback enriches the policy context. In parallel, corrections are distilled into knowledge base $\mathcal{D}_H$, separating device-specific from transferable knowledge.

6.4.3 Transfer Across Devices

When adapting to a new appliance, BookWorm reuses knowledge from prior appliances in the same category. These devices often share task structure, manipulation strategies, and relevant feedback, despite differences in geometry, layout, interface, and articulation.

Given a target appliance D_t, BookWorm transfers the policy π_{pf} from source appliance D_s. The system establishes the target device grounding Γ_{D_t} using observations and the new device's guidebook. The skill library Ω is then initialized from D_s. To account for geometric and morphological differences, BookWorm generates new skill variants using the target guidebook, the interaction history $\mathfrak{D}_H$, and the updated grounding Γ_{D_t}. During execution, the policy dynamically selects among these skill variants based on current observations and the guidebook, minimizing required human feedback on D_t while preserving capabilities on D_s.

6.5 Experimental Setup

6.5.1 Tasks

6.5.1.1 Overview

Figure 6.3: Experimental Appliances by Setting.

We evaluate our method on a real-world Franka Emika Panda Arm in both the *single device* setting and the *transfer learning* setting between multiple devices. Guidebooks are downloaded as PDFs, with only the English-language pages processed for efficiency. For the *single device* setting, we evaluate on a **Honeywell thermostat** (X2P-RTH20B) where the task is to disable schedule settings by navigating multiple menus, and a **Mr. Coffee maker** (PC05) to load a filter, brew coffee, and adjust the time. To evaluate multi-device *transfer learning*,

we experiment with two device pairs. In the coffee maker pair, we first train on Mr. Coffee, then transfer the learned policy to a novel, morphologically distinct **Black-and-Decker coffee maker** (CM1160-W). In the waffle maker pair, we first train on an **Amazon Basics waffle maker** (AB-258NA), then transfer to a **Chefman waffle maker** (RJ04-V3). All tasks involve four discrete skill steps. See Figure 6.3 for our experimental appliances.

6.5.1.2 Task Details and Success Criteria

On the **thermostat** task of turning off the schedule settings, the thermostat is placed in an axis-aligned pose with the screen right-side-up for the wrist camera. The schedule setting starts "on," and the thermostat is on the home screen and idle (i.e., the screen is not lit). A successful end-state of the thermostat is (1) the schedule setting has been turned to "off," and (2) the robot has navigated back to the home screen on the thermostat. To complete the task, the robot must: (1) Press left middle button ("Menu"), (2) Press left bottom button ("Select"), (3) Press right down button (which toggles the setting from on to off), (4) Press left top button twice ("Save & Exit", "Exit").

On the **Mr. Coffee maker**, the task is to brew a cup of coffee immediately, then adjust the time by at least two hours. The coffee maker is set near the center region of the workspace, with its buttons visible to the overhead camera. Initially, the coffee maker is plugged in with its lid open, and the reusable filter is placed a few inches in front and to the right of the console on the coffee machine. Both the "brew now" and "brew later" buttons are off (not illuminated). The goal is to ensure that (1) the reusable filter has been placed inside the brew basket with the lid closed, (2) the time has advanced at least two hours from the time in the initial condition, and (3) the "brew now" button is illuminated. To complete the task, the robot must: (1) Insert the coffee filter into the brew basket, (2) Close the lid, (3) Press the "Brew Now" button, and (4) Press the "HR" button twice.

On the **Black-and-Decker coffee maker**, the task is the same as on the Mr. Coffee coffee maker, as this device is only used for transfer learning experiments. As such, the initial and goal conditions of this task and device are also extremely similar to that of Mr. Coffee Maker. The main difference is that the light indicator after pressing the brew button is located in the black screen, instead of on the button. To succeed, the robot must: (1) Insert the coffee filter into the brew basket. (2) Close the lid. (3) Press the "On/off" button. (4) Press the "hour" button twice.

On the **Amazon Basics waffle maker**, the task is to make a light brown waffle. The waffle maker starts opened, with the dial set to off, and a cup of waffle batter off to the side. The goal is to (1) ensure the waffle has been cooked at light brown setting, and (2) the knob is back to off at the end of the task. To complete the task, the robot must: (1) pour batter into the waffle grill bottom, (2) close the lid, (3) turn the knob to the "light" setting, and (4) turn the knob back to the "off" setting.

On the **Chefman waffle maker**, the task is the same as on the Amazon Basics waffle maker, as this device is also only used for transfer learning experiments. Thus, the initial and goal conditions of this task and device are also extremely similar to that of Amazon Basics waffle maker. The main difference is that the knob must be turned to different angles for the same browning settings. The four steps on this waffle maker are the same as the previous waffle maker.

6.5.2 Training and Evaluation Procedure

For each method on each device, we train for five trials, during which the human can be queried for corrective natural language feedback, before evaluating for ten trials. All trials have a maximum horizon of 12 timesteps, and all methods emit an empty action to terminate the trial. After providing feedback, the human operator manually resets the scene if an irrecoverable failure has occurred, allowing the trial to resume. During evaluation, all modules are frozen and operate without

human supervision. Consequently, the failure detector is disabled, though the policy continues to predict the next state to improve skill and parameter selection.

6.5.3 Metrics

We evaluate three key metrics: (1) full task success, where all steps must be performed in the correct order; (2) step completion rate, the average proportion of consecutive steps completed (out of 4 steps per task); and (3) number of human feedback rounds, to quantify human training effort.

6.5.4 Baselines

6.5.4.1 Overview

In the *single device setting*, we compare against **Code-as-Policies (CaP)** (Liang et al., 2022) and **Promptbook** (Arenas et al., 2024). CaP directly creates skill functions from a guidebook but does not explicitly ground action or dynamics information from the guidebook, nor solicit human feedback. Promptbook is similar to CaP but adheres to more concrete prompt guidelines and *can* query for human feedback. In the *transfer learning setting*, we compare against **Experience Replay** (Rolnick et al., 2019), which is provided the same human feedback as our policy. All baselines are provided the same observations as our method.

6.5.4.2 Baseline Implementation Details

Code-as-Policies (CaP) (Liang et al., 2022). At the beginning of each episode, CaP creates a new skill library in Python. At each timestep of the episode, CaP predicts the skill-parameter call as the action to execute in the world, and an optional skill function redefinition for any $\omega \in \Omega$, which redefines ω for all future timesteps in the episode. Consistent with the original paper, our CaP baseline lacks human-in-the-loop capabilities and is therefore trained without human feedback.

PromptBook (Arenas et al., 2024). The key differences between Code-as-

Policies and Promptbook are the 7 specific ingredients in the prompt of Promptbook, as specified in the paper, and its access to human feedback. Promptbook is permitted to optionally redefine any skill function and query for human intervention upon detecting a lack of progress. At the end of each episode, the human has the opportunity to provide episode-level feedback to Promptbook, which is integrated in the context of the next episode. Because Promptbook supports human refinement, we allow it to maintain the same, continually updating skill functions over each episode, similar to our method.

Experience Replay (Rolnick et al., 2019). Experience replay is a simple and popular approach from the continual learning literature, where data from older tasks are co-trained with data from the current task to prevent catastrophic forgetting. We implement experience replay by injecting an aggregated history of human feedback directly into the policy's context. This history consists of identical feedback to what our method accumulated for both the policy and skill functions during its five training episodes per device. However, experience replay omits the failure-triggered, iterative human interventions that are central to the performance of BookWorm.

6.5.5 Ablations

In the *single device setting*, we test four ablations of our method: not using the guidebook for planning or skill synthesis (**No guidebook**), not predicting the next state with the policy (**No next state pred.**), and not having human language feedback (**No human feedback**). In the *transfer learning setting*, we ablate the value of skills learned and feedback accumulated from the source appliance by training everything from scratch on the target appliance (**Ours, Single Device**).

6.6 Experimental Results

To evaluate our framework, we seek to answer three research questions: (RQ1) Does our method effectively implement and execute actions from guidebooks? (RQ2)

Does our method effectively utilize dynamics information from guidebooks? (RQ3) Can knowledge acquired from previous devices be leveraged to more efficiently learn on new devices within the same category? Figure 6.4 visualizes the results for RQ1 and RQ2 in the *single device setting* (left), and RQ3 in the *transfer learning setting* (right). Across all conditions (10 trials per bar), solid bars denote full task success, while hollow, diagonally-striped bars represent step completion rates. We answer each question below.

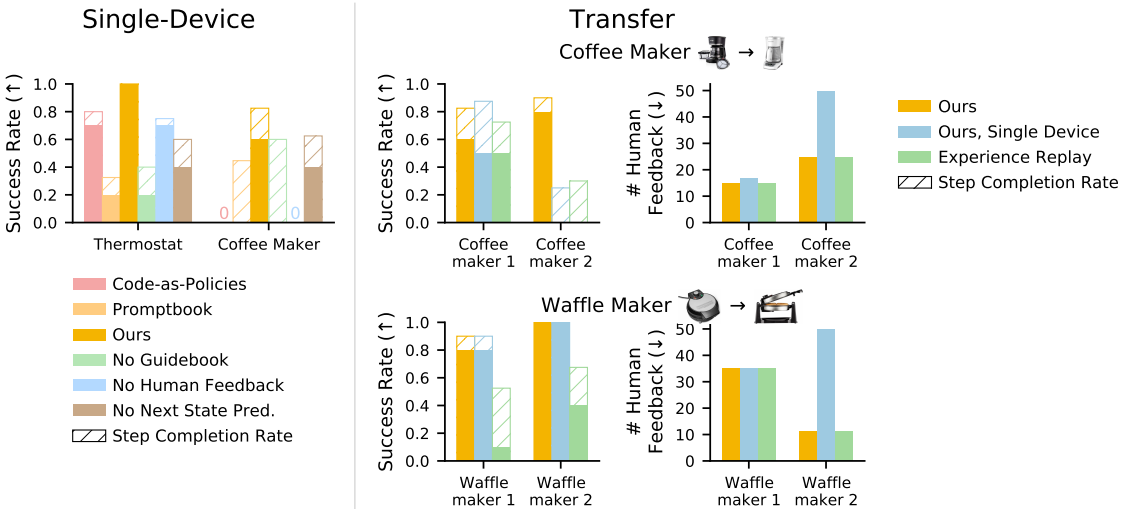


Figure 6.4: **Left**: In the single device setting, our method outperforms prior work baselines and all ablations on both devices. **Right**: In the transfer learning setting, our method achieves comparable success as ablations and baselines when learning the first device, but greatly reduces human effort compared to a single-device ablation learned solely on the second device.

6.6.1 RQ1: Skill execution and implementation from guidebooks

Removing the guidebook from the policy and skill synthesizer (**No Guidebook**) decreases the average single device success rate of BookWorm from 80% to 10%, confirming that guidebook grounding is necessary for both skill synthesis and sequential planning. However, the guidebook alone is not sufficient for enabling performant policies. We also demonstrate the importance of soliciting human feedback

for refining our guidebook-initiated policies. Our method without human feedback only averages 35% over the two devices.

We also compare to prior works CaP and Promptbook to measure how effectively they can formulate policies grounded to guidebooks. CaP and Promptbook achieve 35% and 10% respectively, far lower than ours.

6.6.2 RQ2: Guidebook Dynamics for State Tracking and Failure Detection

In BookWorm, the policy jointly predicts the action and the resulting next state in language. Removing next-state prediction from the policy drops performance from 80% to 40%, demonstrating that BookWorm actively grounds guidebook dynamics in real-world transitions to enable better action predictions.

6.6.3 RQ3: Transfer Learning Across Devices

In the transfer learning setting, our method adapts more efficiently, with fewer rounds of human feedback, to the new appliance than all other methods tested. Our method performs 55 percentage points (pp) better than the **Experience Replay** baseline when averaged over both pairs of devices, despite using the same amount of human feedback. This further demonstrates BookWorm's ability to extract generalizable information from prior devices' guidebooks when transferring to new devices. Compared to our single-device ablation that does not transfer knowledge between devices, BookWorm achieves, on average, a 40 pp higher success rate on the target device with 50% to 78% less rounds of human feedback. This demonstrates that our transfer mechanism, which distills generalizable language feedback and refines skill functions from the previous device, significantly improves transfer to new devices within the same appliance class while reducing device-specific human supervision.

6.6.4 Common Failure Modes of BookWorm

On the **thermostat**, common failure modes during training include pressing the button too lightly so that it doesn't register, pressing the button too hard so that a single press registers as multiple presses, and pressing a button after the screen has timed out. Because our VLM-based policy takes several seconds to act, the screen often resets before the skill executes, applying the correct action to the wrong state.

On the **Mr. Coffee maker**, common failure modes include spatial imprecision when pressing buttons and failing to terminate the episode promptly. BookWorm would often continue pressing the "hour" or "brew now" buttons even after completing the task, sometimes undoing its own progress and turning off the coffee maker in the process.

On the **Black-and-Decker coffee maker**, BookWorm primarily struggles with the low-level control required to close the lid. This action is highly sensitive to the appliance's initial pose; slight positional variations often leads to inverse kinematics (IK) failures at prescribed skill waypoints, which inadvertently shifts the machine during execution. It is also very sensitive to the precise waypoints and robot pose, and being a few centimeters off often resulted in skill failure. Similar to the other devices, spatial imprecision during button presses constituted the remaining failure cases.

On the **Amazon Basics waffle maker**, BookWorm most commonly fails when terminating too early (immediately after the lid has closed). BookWorm did have any recorded failures during evaluation on the **Chefman waffle maker**.

6.6.5 Representative Human Feedback

We list some representative examples of human feedback on three sets of experiments.

Thermostat. Task: Turn off the schedule settings.

- "When encountering unfamiliar screens, press 'save and exit' or 'back' to return to a more familiar screen."
- "When button pressing, press 20% harder."
- "Move 0.5cm in the +y direction when pressing the top left button."

Mr. Coffee Maker. Task: Brew a cup of coffee and adjust the time.

- "Try placing the filter into the brew basket again. You hit the coffee maker when trying to put the coffee filter in, leading to it not getting in. Try to get the filter higher (+z direction) before putting the filter in."
- "To close the lid, do not open then close your gripper. The lid does not open about the y axis, it opens about the x axis. Keep the gripper closed, move to a waypoint to the left of the lid (-y direction), then to a waypoint to the right of the lid (+y direction)."
- "Move 0.7cm more forward when pressing buttons."

Black-and-Decker Coffee Maker. Task: Brew a cup of coffee and adjust the time.

- "Open the gripper after the robot reaches the waypoint right above the brew basket. Do not move to a waypoint lower inside the brew basket."
- "When closing the lid, increase your last two waypoints by roughly 5cm in height."
- "Change your button pressing offset to be 1.5cm further to the left."

6.7 Conclusion

We presented BookWorm, a framework for learning real-world robot manipulation tasks from guidebooks by grounding entities, synthesizing executable skills,

and using targeted human feedback to refine and transfer the resulting policy. We empirically showed that our method better utilizes guidebook information than prior works and greatly decreases the amount of human effort needed to learn new devices within the same class in a transfer learning setting. We believe BookWorm opens the door to leveraging the large corpus of human-written domain information to help robots more efficiently learn physical tasks in real households, and that it can substantially improve sample efficiency over robot learning frameworks that do not use guidebooks.

6.7.1 Limitations and Future Work

We experimented with only code-based skill representations. While this representation allows for easy extraction from language guidebooks and refinement with language feedback, it has limited expressivity for articulated appliances where precise waypoints and end-effector orientations are needed. Future work can examine having a hybrid skill library of code-based and learned policy heads to further bridge the gap between textual guidebooks and low-level control. Actions like pick-and-place can be easily represented as code-based skills, while precise and contact-rich insertion tasks may require RL-trained skills.

Furthermore, BookWorm primarily considers how guidebook information can influence skill synthesis, planning, and dynamics understanding. Future work can further exploit guidebook information to learn symbolic reward functions for better progress monitoring and action prediction, or to initialize safety-oriented constraints to help robots avoid hazardous situations cautioned against in guidebooks.

Chapter 7: Future Work

7.1 Language for Faster Learning and Transfer

7.1.1 Language Priors for RL

In DeL-TaCo (Ch. 3) and Lang4Sim2Real (Ch. 4), we explored how language can facilitate transfer to new tasks and new domains with multitask imitation learning policies. While we used a Q-learning framework inspired by temporal-distance RL in MICoBot (Ch. 5), we did not actually experiment with how language can enable RL to learn faster. Since the primary bottleneck of real-world RL is sample efficiency, leveraging language to accelerate RL is one of the most potent areas for future work. While prior work (Goyal et al., 2019, 2020; Yu et al., 2023a; Ma et al., 2024a; Torne et al., 2026) has explored how to use language to shape reward functions that expedite RL, another less-explored area is using language to shape policy and behavioral priors for more efficient, semantically meaningful exploration.

Standard RL in continuous action spaces typically parameterizes the policy as a Gaussian distribution over actions. Sampling actions from a Gaussian policy during exploration is inefficient, as this distribution typically produces random, noisy movements rather than semantically meaningful behavior. To improve exploration efficiency, we replace the Gaussian with a multimodal action distribution where each mode represents low-level actions conditioned on a distinct language instruction. This expressivity enables the policy to capture a wide range of semantically distinct behaviors. Here, language acts as an intermediate action space L_a bridging the high-level (π_{hi}) and low-level (π_{lo}) policies.

We consider the following potential formulation. π_{hi} predicts a categorical distribution over freeform, language-specified actions that elicit semantically diverse behaviors, such as picking up, pouring, or sliding different objects on the scene. π_{hi} outputs action probabilities that dictate the probability mass on each of the modes of

the continual multimodal action distribution outputted by π_{lo}. Concretely, for state space S, language action space probability simplex $\Delta(L_a)$, and low-level action space probability simplex $\Delta(A)$,

$$\pi_{hi} \colon s \to \Delta(L_a) \tag{7.1}$$

$$\pi_{lo} \colon s \times \Delta(L_a) \to \Delta(A) \tag{7.2}$$

To represent a multimodal action distribution with a flexible number of modes, π_{lo} can be instantiated as a flow-based model (Dinh et al., 2015, 2017; Rezende and Mohamed, 2016), an architecture proven to enable faster, semantically meaningful exploration (Singh et al., 2021). By conditioning the low-level action distribution $\Delta(A)$ on the higher-level language-based action distribution $\Delta(L_a)$, the proposed policy can likely produce more expressive, semantically-informed exploration priors than what standard RL provides.

7.1.2 Language for the IL-RL paradigm

Because online RL from scratch is typically intractable for tasks more complex than pick-and-place, a prevailing paradigm for acquiring spatially precise policies is to initialize a base policy via imitation learning (IL) before fine-tuning it with online RL (Rajeswaran et al., 2018). Similarly, offline-online RL algorithms like AWAC (Nair et al., 2021a) and Cal-QL (Nakamoto et al., 2024) perform offline RL (effectively, advantage-weighted imitation learning) to acquire an initial policy before online RL.

However, these learning paradigms assume access to a sufficient amount of demonstrations. Acquiring this data is often prohibitively difficult for highly complex systems, such as quadrupedal robots with onboard manipulators that require whole-body teleoperation. How can language and VLMs act as semantically informed action priors to better guide downstream RL exploration?

In situations where robot demonstrations are infeasible to teleoperate, we propose a Language-based Behavior Prior → Human Feedback → RL learning paradigm.

Initially, we use (1) language-guided high-level reasoning priors, such as LLM/VLM code-based skills (Liang et al., 2022; Huang et al., 2023) or directional cues for navigation (Nasiriany et al., 2024), that are distilled into a policy representation that is then (2) updated with human feedback and (3) finetuned with RL to have millimeter-level precision downstream.

Human feedback in (2) should be multimodal, similar to BookWorm (Ch. 6), where language is used for high-level corrective feedback and keypoints for fine-grained corrections and grounding. One of the key research questions in developing this new learning paradigm is how to create a policy representation that can effectively incorporate human language feedback in step 2 and then be efficiently finetunable in step 3. Representations based on Code-as-Policies (Liang et al., 2022) facilitate step 2 but are not the best choice for step 3 without some factorization, such as formulating the RL action space to be the value of a few waypoints in the code. A straightforward option is to distill the policy representation acquired after steps (1) and (2) into an initial demonstration dataset, on which offline and then online RL can be performed to acquire a spatially precise policy.

7.1.3 New Forms of Natural Language for Robot Learning

In this thesis, we considered language task descriptions with DeL-TaCo (Ch. 3), scene descriptions with Lang4Sim2Real (Ch. 4), task allocation dialog with MICoBot (Ch. 5), and human corrections with BookWorm (Ch. 6). While other categories of language were not investigated in this thesis and remain largely absent from the broader literature, they hold significant potential to unlock performance and sample efficiency gains for robot learning.

7.1.3.1 Task analogies and symmetries

Language task instructions only provide a signal for how similar two tasks are based on their distance in language embedding space. We propose using language

analogies to teach robots new tasks more efficiently based on their understanding of previous tasks. For instance, if one wants to teach a robot how to sprinkle salt on a dish, and the robot already knows how to pour water and shake a bottle, one can say "putting salt on a dish is similar to pouring while shaking the salt container," which leverages the analogy that `put_salt` $\approx$ `pouring` + `shaking`. Humans teach each other through many such analogies, which we expect to also help when teaching robots. This extends the idea of vector space arithmetic in NLP (Levy and Goldberg, 2014; Ethayarajh et al., 2019; Allen and Hospedales, 2019; Mikolov et al., 2013), based on the classic `king` $-$ `man` $+$ `woman` $=$ `queen` analogy.

Just as semantic analogies can be leveraged to learn new skills, geometric symmetries can be exploited to efficiently learn physical behaviors. Recent robot learning literature leverages inherent geometric symmetries to train sample-efficient equivariant models (Wang et al., 2022c,b; Zhu et al., 2025b). While these works do not leverage language, future work should explore how to teach *approximate* 3D-space symmetries through language to a robot learning model. For instance, a stirring motion in a pot is similar to a circular wiping motion on a table, when we view the path of the hand in the xy plane. The key research question is how to leverage language-specified analogies and symmetries across multiple verbs to learn new motions. This is particularly challenging when applying these verbs to different scene objects, which prevents directly transferring trajectory paths.

7.1.3.2 FAQs and common trouble-shooting advice

Robots fail often while learning. To mitigate failure modes, researchers usually use DAgger (Ross et al., 2011) to collect expert data on recovery from out-of-distribution states, or refine the policy with human-in-the-loop language feedback. We propose an area of future work that leverages self-diagnosing knowledge bases and trouble-shooting guides online. For instance, on wikihow and user's manuals, there are FAQs and common trouble-shooting advice that are useful to humans performing a long-horizon task. Research in this area must address two core questions:

(1) how to ground textual and diagrammatic error symptoms to the robot's visual observations, and (2) how to translate high-level troubleshooting advice into continuous control actions.

7.1.4 Cross-Embodiment Learning with Language

In DeL-TaCo (Ch. 3) and Lang4Sim2Real (Ch. 4), we explored how to use language-shaped representations to efficiently transfer policies to new tasks and environment domains. We expect a similar language-shaped, domain-invariant representation to effectively facilitate morphological transfer, helping adapt skills across different robot platforms or translate human video demonstrations into robot actions.

Another promising direction for cross-embodiment learning lies in using language as a mid-level action representation for a hierarchical policy, similar to RT-H (Belkhale et al., 2024) and YAY-Robot (Shi et al., 2024). In these works, the policy is factorized into a high and low level. The high-level emits tokens of natural language actions, such as "move your arm left," and the low-level policy converts the language tokens into continuous space end-effector control. While RT-H and YAY-Robot experimented with a single morphology, their policy factorization is promising for cross-embodiment learning because the high-level policy can serve as a morphology-agnostic backbone while the low-level policy is trained to be morphology-specific. The challenge here lies in determining the appropriate granularity of this mid-level language action representation, since a robot's morphology can still impact the feasibility and optimality of the mid-level language action sequence. Furthermore, the high-level policy must learn representations general enough to facilitate cross-embodiment transfer without sacrificing the adaptability of the low-level policy to new morphologies.

7.1.5 Learning Complex Reward Functions from Language

Future work can build upon language-guided reward functions (Goyal et al., 2019, 2020) and its extensions to LLM code-written reward functions (Yu et al., 2023a; Ma et al., 2024a). Specifically, we can expand on recent work (Torne et al., 2026) that leverages freeform language to define human preference axes and then trains reward models to guide upside-down reinforcement learning (Schmidhuber, 2020) for vision-language-action policies.

Some user preferences are spatial ("stay far from the laptop") while others are more temporal and behavioral ("move slower when you are around people"). Different preferences may require different reward function representations. Can we use a mix of VoxPoser (Huang et al., 2023)-like rewards (which are voxel-based reward landscapes) for user preferences that involve spatial precision, and rewards based on latent trajectory representations for temporal and behavioral user-preferences?

7.2 Language for Smoother HRI

In MICoBot (Ch. 5), we described a system for human-robot collaborative manipulation with flexible, bidirectionally-initiated dialog, and in BookWorm (Ch. 6), our guidebook-following robot incorporated on-demand human corrections to update its skills. We propose future work directions that combine both of these threads.

7.2.1 Human-robot collaboration with Mixed-initiative Teaching

MICoBot (Ch. 5) cannot learn new skills or improve its success rate on existing ones based on collaborative experience. Consequently, even with a consistently helpful user on a repeated task, the robot's reliance on human assistance never decreases. While MICoBot (Ch. 5) adapts to the human's helpfulness within a single trial, practical household deployment requires a collaborative robot that continually improves its skills and adaptiveness to human preferences across days and weeks.

Creating a continually learning collaborative robot raises a few interesting research challenges. First, when the human executes task steps that exceed the robot's current capabilities, how should onboard sensor data be used to capture human physical actions to facilitate subsequent skill learning? Second, how can the robot leverage its physical and verbal action spaces to actively guide human demonstrations—such as requesting specific initial or final object configurations—to collect more diverse training data? Third, what criteria should the robot evaluate to decide when to initiate a request for the human to perform the next step, to collect this data? Fourth, when should the robot take the initiative to demonstrate a skill to the human, either to adjust the environment for a seamless handoff (e.g., placing objects in specific configurations) or to teach the human novel skills, such as operating an unfamiliar appliance? More fundamentally, how might optimizing how a robot teaches humans simultaneously improve how it learns from them?

7.2.2 Real-time Simultaneous Collaboration

In factory and home settings, many tasks cannot be done physically by either agent alone, such as moving large pieces of furniture. During MICoBot (Ch. 5) user studies, a participant asked the robot to cut a piece of tape while they peeled it away. In these scenarios, simultaneous co-manipulation requires real-time spatial awareness, continuous and bidirectional physical feedback, and explicit verbal coordination regarding agent trajectories and object states (e.g., "While I lift the from two legs of the table, pull the carpet out," or "Angle it to your left through the doorway, and I will move slightly to my right"). Mixed-initiative dialog is essential in these situations.

How can we extend MICoBot (Ch. 5), which operates on a predetermined plan, to this real-time, simultaneous human-robot collaboration setting? First, both agents must have a open-world, real-time perception pipeline, enabling the robot to constantly update its state representation of the world. It must track the 3D scene and relative pose of task-relevant objects. Second, to physically coordinate with the user in real time and anticipate their movements, the robot needs a short-horizon

forecasting model of human trajectories (similar to MOSAIC (Wang et al., 2024)). This human model should be conditioned on the previous dialog history and 3D scene representations. Third, it must decide whether to initiate dialog to influence this forecasted human motion trajectory and when it should communicate its own trajectory plan.

A key technical challenge lies in enabling intra-step initiative shifts. Unlike MICoBot (Ch. 5), where initiative shifts only occur after task steps are completed, the new system must fluidly accommodate initiative shifts during the execution of individual steps. This demands a highly responsive mechanism for seizing and ceding initiative. One possible trigger for seizing initiative would be to mitigate safety risks, such as preventing imminent collisions outside the human's field of view. Upon assuming the lead, the system must plan out, spatiotemporally synchronize its actions with the user's, and generate concise verbal dialog to facilitate this physical coordination. Upon ceding initiative, the robot must seamlessly transition into a compliant, instruction-following system.

Building such a real-time system involves drastic time complexity improvements over current state-of-the-art VLMs. In MICoBot (Ch. 5), system latency is heavily bottlenecked by computation: the meta-planner requires ≈ 15 seconds per call, while the mobile manipulation primitives require ≈ 20 seconds for segmentation and object selection alone, before physical execution occurs. While serviceable for controlled user studies, this latency is too high and lacks responsiveness for practical household collaboration tasks like moving furniture.

To improve sample complexity, we can incorporate three advances from the speech and language community. First, we can deploy a hierarchical System 1/System 2 architecture, where a lightweight model handles immediate, reactive dialog, while complex queries are routed to a slower, larger reasoning model that runs in the background (Lab, 2026; OpenAI, 2026). Second, we can enable full-duplex communication (Défossez et al., 2024), allowing the robot to process human interruptions and

environmental audio simultaneously while speaking to the human and acting physically. Finally, we can apply anticipatory generation to the robotics setting, where transformer-based models learn to initiate computation before the human finishes their query, saving several seconds of latency (Shih et al., 2025).

7.2.3 Improving Human-Robot Co-Adaptation

In MICoBot (Ch. 5), we saw how bidirectionally-initiated dialog can greatly enhance human-robot collaboration. During the course of collaboration, both agents continually adapt to each other. For instance, MICoBot (Ch. 5) is designed to continually adapt to the human's preferences and willingness to help over time, and the human may also quickly develop new strategies for working with the robot. We call this phenomenon co-adaptation, also known as entrainment in the dialog systems community. However, we did not study the human side of co-adaptation and entrainment. For instance, we observed that users quickly understood the limits of robot capabilities, which caused some users to give up on collaborating with the robot for chunks of the task that they pre-judged it unable to perform. In one user study, impatience with the robot's speed caused one participant to perform the entire rest of the task. To mitigate this drop in collaboration and improve the robustness of MICoBot (Ch. 5) to human behavioral changes, future research must examine mixed-initiative teaching (Sec. 7.2.1) and low-level motion feedback integrated into the dialog loop. These additions would enable continuous mutual improvement: humans could interactively correct suboptimal policies, while the robot provides feedback about initial condition constraints under which it can perform skills accurately, thereby preventing the break in collaboration observed in our studies.

Studying human-robot co-adaptation would require within-subjects studies over several weeks that explore different facets of continuous mutual adaptation during collaboration (Schneiders et al., 2024; Argall and Billard, 2010) beyond MICoBot (Ch. 5). Specifically, future researchers could examine: (1) **Initiative shifts**: the

collaborative dynamics by which agents take and cede initiative and control. As observed in our prior studies, asymmetric adaptation can lead to undesirable extremes, such as the human assuming total control and obviating the robot's usefulness. (2) **Physical synchronization**: how agents align their physical movements during simultaneous actions (Sec. 7.2.2), such as collaborative transport or handovers. The success of this synchronization is intrinsically tied to which agent currently holds the initiative. (3) **Multimodal communication**: expanding the robot action and observation space to perceive and produce auditory and haptic cues to establish mutual awareness of status and progress with the human collaborator. To become an effective partner, the robot must learn to actively and efficiently signal its state, such as through auditory cues to alert the human when it requires a physical demonstration or verbal correction to resolve an error. Similarly, haptic feedback during simultaneous manipulation must adapt over time; for example, a robot handing over an object should adjust the timing of its gripper release and haptic confirmation to match the human's increasingly efficient grasp patterns as they acclimate to the robot.

7.3 Defining Categories in which Language Aids in Generalization and Transfer

We use language to transfer between multiple semantic categories—tasks (DeL-TaCo (Ch. 3)), domains (Lang4Sim2Real (Ch. 4)), and devices (BookWorm (Ch. 6))—but each work contains many underlying assumptions about the structure and definition of each semantic category. In DeL-TaCo (Ch. 3), we assume that the test tasks must be semantically similar to the training tasks so that the language task embeddings are sufficiently within distribution during testing. Empirically, each task in our pick-and-place benchmark was parameterized by the picked object and its place location. This definition of "task" was chosen quite narrowly for language to aid in one-shot generalization—the policy implicitly learned what pick-and-place motions looked like, and it did not have to generalize to a new task outside of this family

of motions. In Lang4Sim2Real (Ch. 4), we assume that (1) the simulation and real domain data contain semantically similar tasks, and (2) there exists a way to divide the trajectories between sim and real into the same number of stages, where within each stage across both domains, the policy must predict a similar distribution of actions. However, the key questions are how semantically similar the simulated and real-world tasks must be, and how closely the actions within each stage must align across domains. In BookWorm (Ch. 6), we assume that there exists a notion of a device class, within which we can effectively transfer our policy from one device to another, but we sidestepped the fundamental problem of defining a class within which useful transfer could occur.

Because language provides semantics that can be generalizable from one situation to another, future work should characterize what defines the categories within which language can aid in generalization, and whether these categories are hard or soft labels. How might one characterize notions of tasks, domains, and device classes?

One potential way is to rely on a continuous similarity metric between any two entities, and threshold it so a pair of entities above a certain similarity score are considered to be part of the same semantic category. For instance, task similarity can be defined as a learned language similarity metric (e.g., cosine distance on output embeddings of LLM-embedding models) between the two task descriptions, trajectory similarity score between the mean demonstration of each task (e.g., dynamic time warping), or a combination. For longer tasks, it can be defined recursively by averaging the similarity of constituent steps in the task. Domain similarity can be defined through the combination of scene segmentation model embeddings or dense language scene description embeddings. If there are cluttered scenes, the segmentation and language descriptions can be filtered with off-the-shelf visual reasoning models to only include task-relevant objects. Finally, device similarity can be evaluated along two axes: functional semantic similarity (derived from dense query representations of the device's human-designated purpose) and geometric similarity. To capture the latter, future work could utilize articulated 3D device embeddings—produced by a

pointcloud encoder trained on PartNet (Mo et al., 2019) from Sapien (Xiang et al., 2020)—ensuring that devices with similar actuation axes and kinematics are embedded in close proximity, regardless of their semantic labels.

By using these similarity metrics for tasks, domains, and devices, and extending them to additional semantic categories, we can create a custom categorization such that two entities are members of the same category if they have high enough similarity.

7.4 Language-conditioned World Models

In BookWorm (Ch. 6), we demonstrated that textual dynamics extracted from guidebooks improve policy predictions. However, in safety-critical settings like operating electrical appliances, ensuring human safety requires simulating the physical consequences of proposed actions so that unsafe trajectories can be discarded and the policy resampled prior to execution. Visual world models (Ha and Schmidhuber, 2018) promise to enable this simulation by using the same autoregressive objectives that powered the generalization capabilities of LLMs. However, current visual world models lack physical and semantic grounding (Thorpe et al., 2026). They hallucinate physically impossible events and fail to maintain temporal consistency under occlusions (Hansen and Wang, 2026).

We propose using language descriptions as symbolic state representations to ground visual world models. We envision a language-conditioned visual world model inspired by Dreamer (Hafner et al., 2019, 2021, 2023, 2025), which maintains a latent state representation z_t that is decoded into next frames. In our setting, we would also decode language from z_t to increase the faithfulness of image reconstructions to the underlying symbolic state of the world. Additionally, the reward prediction head can be conditioned on z_t, because reward functions are often based on symbolic states of the world (e.g., whether the object been placed into the plate).

How can we leverage textual dynamics descriptions from guidebooks or the

internet to supplement the prediction of this language-augmented latent z_t? The key challenge lies in aligning the high-level granularity of textual dynamics descriptions (e.g., "after pressing the button, the green indicator turns on") to the low-level state and action granularity of the world model. Works like THICK (Gumbsch et al., 2024) tackle different levels of temporal abstraction by creating a dual-timescale world model, where the lower level predicts fine-grained dynamics at the robot's action frequency, and the higher level operates over sparsely updated latents to predict discrete state changes over time. However, THICK learns purely from visual data, so it does not consider how to align visual predictions with the semantic timescales and milestones described by textual descriptions. Consequently, the open challenge is to develop world models that explicitly anchor these unsupervised visual transitions to textual dynamics from guidebooks, ultimately enabling rich textual information to guide the prediction of the language-augmented latent z_t.

7.5 From Appliance Manuals to Behavioral Principles

Users expect a household robot to not only follow instructions and learn efficiently from textual information, such as guidebooks (as we presented in BookWorm (Ch. 6)), but to always comply with certain behavioral principles that ensure safety, utility, and adaptability to human preferences. Analogous to personalized system prompts (Zhang et al., 2018) and constitutional AI (Bai et al., 2022b) for LLMs, a natural extension of BookWorm (Ch. 6) is to ground the robot's execution in high-level operational principles that dictate overarching safety and human-alignment rules.

How can we convert written user preferences into motion-level feedback for robot training? Should this feedback be represented as cost/reward landscapes, or in some other numerical form, for behavioral optimization? How can this system learn to obey a hierarchy of rules and user preferences (Censi et al., 2019b), given the robot's physical capabilities, to synthesize safe actions? For example, consider two rules: (1) do not break fragile items, and (2) keep the dining table clean, where (1)

precedes (2) in importance. If the robot knows it cannot safely grasp a delicate wine glass on the table, attempting so risks violating rule 1 in service of following rule 2. By reasoning about both the consequences of its actions and its capability limits, the robot can formulate an alternative plan, such as sliding the glass onto a serving tray it can safely hold, satisfying both rules. When principles conflict, the robot must learn to prioritize those at the top of the hierarchy.

For **safety principles**, we noted earlier that BookWorm (Ch. 6) did not leverage the large amount of safety information and constraints in guidebooks. We note that safety constraints can roughly be categorized into state-constraints (e.g., "it is bad for your gripper to be within 2 cm of the tabletop"), action-constraints (e.g., "always stay under 5mph"), and state-action-constraints (e.g, "do not touch the stove if it is hot", or "release the valve immediately after cooking has finished"). Similar to the temporal abstraction challenges mentioned in Sec. 7.4, a key challenge is grounding these abstract, language-described states and actions into the state and action space of the robot.

Unlike MICoBot (Ch. 5), BookWorm (Ch. 6) does not consider how to adapt to **human preferences** specified in language. These can range from high-level task constraints (e.g., strictly separating raw meat and vegetable utensils, or pairing specific foods with specific sauces) to low-level spatial arrangements (e.g., storing chopping boards vertically instead of flat, or ensuring knives face inward while chopsticks face outward).

So far, we have focused our discussion on enabling a robot's physical behavior to adhere to user-specified principles in language. It is imperative that these constraints extend beyond physical execution to govern **verbal actions and dialog interactions** as well. By unifying the dialog capabilities of MICoBot (Ch. 5) with the operational rule-following capabilities of BookWorm (Ch. 6), we can ensure the robot's speech remains as safe and strictly compliant as its physical movements. Unlike issues of safety in LLM research, in robotics, the verbal actions of a robot must

be grounded in its physical actions and the current physical state of the world. For instance, a robot that has just broken and is starting to clean up some glassware on the kitchen floor should verbally warn an approaching toddler to stay clear of the glass. In such cases, initiating verbal actions serves as a critical mechanism for rule compliance, satisfying overarching mandates like keeping humans safe from hazards. The core research challenge lies in operationalizing high-level, language-specified principles into a unified policy of physical and verbal behaviors, particularly when speech is used proactively to influence the physical environment and uphold user mandates.

7.6 Further directions to scale this thesis to household conditions

7.6.1 Updated Architectures

DeL-TaCo (Ch. 3) and Lang4Sim2Real (Ch. 4) used simpler architectures with ResNet visual backbones (He et al., 2015) that predate VLAs (Brohan et al., 2022, 2023), today's most capable policy architectures. While we validate these proof-of-concept ideas on simpler architectures to establish the value of language for generalization, future work must scale these insights to modern transformer and VLA architectures, similar to how MUTEX (Shah et al., 2023) built upon foundational ideas from DeL-TaCo (Ch. 3).

7.6.2 Internet-scale Pretraining

While DeL-TaCo (Ch. 3) and Lang4Sim2Real (Ch. 4) used pretrained *language* representations to achieve language-based generalization to new tasks and domains, these works failed to leverage the power of large-scale robotic datasets and pretrained *visual* representations to achieve a higher degree of generalization to new tasks and domains.

In DeL-TaCo (Ch. 3), we demonstrated that policies conditioned on both language and demonstration embeddings from a CLIP (Radford et al., 2021) encoder

improved over demo-only or language-only policies. However, CLIP failed to outperform our lightweight, trained-from-scratch encoder. This may be because CLIP was trained on internet-scale image-captioning data that yielded object-centric representations lacking the spatiotemporal understanding and awareness needed for robotics, and that CLIP representations do not generalize well to robot simulation images.

In Lang4Sim2Real (Ch. 4), we experimented starting from an R3M (Nair et al., 2022) visual encoder to compare its performance to a trained-from-scratch ResNet. R3M has better temporal understanding than CLIP embeddings because they are trained on video data with a time-constrastive loss term. However, we found that finetuning R3M representations on Lang4Sim2Real (Ch. 4) objectives yielded a worse policy than off-the-shelf, unmodified R3M representations. We hypothesize that Lang4Sim2Real (Ch. 4) perturbs the pretrained R3M weights too much, losing the benefit of the large-scale pretraining behind R3M. Since it is often infeasible to cotrain with large-scale pretraining data when doing Lang4Sim2Real (Ch. 4) pretraining, future work can examine how to simultaneously optimize for both the pretraining and Lang4Sim2Real (Ch. 4) objectives. This might involve applying a KL divergence penalty, analogous to the trust-region constraints in TRPO (Schulman et al., 2017a) and PPO (Schulman et al., 2017b), to ensure auxiliary task logits do not drift excessively from the original pretrained logits. We also found that augmenting R3M with small, trainable linear adapters (i.e., LoRA (Hu et al., 2021)) yielded significantly better real-world performance than fully fine-tuning the entire backbone during policy training. Future work should explore combining these adapters with the previously discussed KL divergence constraint to further mitigate representation drift, preserving large-scale pretraining benefits while effectively optimizing the Lang4Sim2Real (Ch. 4) objective.

In MICoBot (Ch. 5) and BookWorm (Ch. 6), we utilized pretrained image segmentation models for grounding and VLMs for policies, but did not incorporate large-scale robotic datasets to enhance manipulation. Future work should extract robust priors from diverse sources, including analytical grasping datasets (e.g., Grasp-

Net (Fang et al., 2023, 2020), DexNet (Mahler et al., 2016, 2017)), simulated grasping datasets (e.g., ACRONYM (Eppner et al., 2020)), and large-scale trajectory datasets (e.g., DROID (Khazatsky et al., 2024), BridgeData V2 (Walke et al., 2023)). For BookWorm (Ch. 6), GraspNet could serve as an API call, allowing semantic code-based skills like `insert` to query optimal grasp poses dynamically. Furthermore, BookWorm (Ch. 6) currently requires manual annotation for difficult-to-segment articulated parts like buttons, lids, and compartments. Training the underlying vision models on simulated assets with ground-truth part segmentation masks like PartNet-Mobility (Mo et al., 2019; Xiang et al., 2020) could significantly improve zero-shot grounding precision and reduce human keypoint supervision.

Chapter 8: Conclusion

To deploy robotic manipulation systems to unstructured homes, robots must learn to act intelligently based on behaviors extracted from large amounts of data. However, real robot data is scarce and expensive to collect. We argue that natural language is an abundant and powerful data modality to augment robotic capabilities for two reasons: (1) language is a store of semantic meaning important for a robot to generalize to new domains and tasks, and (2) language forms the basis of human-robot communication and collaboration. We demonstrated several promising results that leverage both benefits of language.

In Chapter 3, we introduce DeL-TaCo, a task-conditioning architecture evaluated on a new benchmark of hundreds of pick-and-place tasks. By fusing visual demonstrations with natural language instructions, DeL-TaCo outperforms policies trained on a single modality. We show that providing just one language instruction provides the same performance gains as fine-tuning on 50 visual demonstrations. Furthermore, the architecture cross-references both modalities to resolve mutual ambiguities. DeL-TaCo achieves significant sample-efficiency gains by leveraging language as a store of semantic meaning, which enables robust transfer to novel tasks.

Building on this concept, Chapter 4 demonstrates that language also enables efficient transfer to novel domains. We introduce Lang4Sim2Real, a lightweight vision-language pretraining method for sim2real transfer of visuomotor policies. This method leverages language to ground visually dissimilar but semantically similar images across simulated and real environments. Consequently, the resulting domain-invariant representations are robust to wide sim2real gaps involving altered camera angles, varying scenes, and differing physical parameters. Ultimately, our approach improves real-world task success rates by 20-30 percentage points over sim2real approaches and leading vision-language internet-scale pretrained representations, even on tasks involving coarsely simulated, deformable objects such as wire wrapping.

We then transition to the second benefit of language: enabling effective human-robot communication. In Chapter 5, we described MICoBot, the first system that integrated mixed-initiative dialog with HRI for real-world, long-horizon collaborative manipulation tasks. We argue that mixed-initiative dialog creates more adaptive teams capable of accommodating varying levels of robot skill capabilities and dynamically changing human helpfulness. To achieve this, the system allocates future steps using an RL-inspired temporal-distance metric that balances task success and physical effort, grounding all subsequent dialog in this optimally computed allocation. In user studies involving 18 participants, MICoBot facilitated significantly more initiative shifts per episode (2.4) than the pure LLM baseline (1.1). Furthermore, the system was highly effective at negotiating with humans to secure assistance on steps it was less competent at. Together, these capabilities enhanced the adaptability of the human-robot team, boosting the success rate by 50 percentage points over the baseline and enabling it to be preferred by 78% of participants.

Finally, Chapter 6 unifies both benefits of language by introducing BookWorm, a framework that leverages textual user manuals to efficiently learn how to operate common household devices. Utilizing the first benefit of language, BookWorm extracts semantic skill priors from these guidebooks to generate a code-based representation that transfers efficiently to new devices. To refine these skills, the framework then leverages the second benefit of language—building upon the robot-initiative dialog capabilities introduced in MICoBot—by soliciting and incorporating natural language feedback from humans. For single-device learning, BookWorm yields a 55-percentage-point success rate improvement over Code-as-Policies. In multi-device transfer settings, it improves target device performance by 65 percentage points, requiring 50% fewer rounds of human feedback than an experience replay baseline. These results demonstrate BookWorm's ability to extract robust, guidebook-grounded policy representations that adapt to semantically related devices with significantly reduced human feedback effort.

Ultimately, the frameworks presented in this thesis establish natural language

as a uniquely powerful modality for enabling open-world robotic manipulation. By harnessing language as a rich store of semantic meaning, we accelerate policy generalization across novel tasks, domains, and devices. By employing language as a communicative medium, we enable efficient learning from human feedback and foster fluid human-robot collaboration in unstructured environments. Building upon these foundations, Chapter 7 outlines promising avenues for future research, such as leveraging language to accelerate exploration in reinforcement learning and tapping into richer forms of language beyond instructions, scene descriptions, and dialog. Pursuing these directions will further drive the sample-efficiency and generalization gains necessary to deploy truly capable robotic helpers to *any* unstructured human environment.

Appendix A: DeL-TaCo Details

A.1 Success Rate Calculation Details

We run each setting with three random seeds for 800k-900k training steps. An evaluation set, which we define as rolling out the policy for 2 trials per task for all of the test tasks, is run every 10k training steps. Thus, there are a total of 80-90 evaluation sets that occur throughout training. Let seed i attain the success rate $r(i, j)$ on evaluation set j. Let J be the top 10 evaluation set indices for the quantity $\frac{1}{N}\sum_i r(i, j)$. Our reported success rate and standard deviation in the tables are calculated as the following equations:

$$\text{Reported Success Rate} = \text{mean}_{j \in J}(\text{mean}_i r(i, j))$$
$$\text{Reported Standard Deviation} = \text{mean}_{j \in J}(\text{stddev}_i r(i, j))$$

In Scenario A, since there are 102 test tasks, each success rate in Tables 3.6 and 3.8 is computed from:

$$\frac{\text{2 trials}}{\text{test task}} \times \frac{\text{102 test tasks}}{\text{evaluation set}} \times \frac{\text{10 evaluation sets}}{\text{seed}} \times 3 \text{ seeds} = 6120 \text{ evaluation trials}$$

Applying the same calculation for the 54 tasks in Scenario B, each success rate in Table 3.7 is computed from 3240 evaluation trials.

For Table 3.8, the best final checkpoint of the three demo-only policy seeds from Table 3.6 was taken for finetuning.

A.2 Scripted Policy Details

We collect $26,000 - 31,000$ training demonstrations using a scripted policy with Gaussian noise added to the action of each timestep. Details are shown in

Algorithm 3. "eePos" stands for end-effector position. All variables ending in "Pos" are xyz positions. Our training buffer contains only successful demonstrations.

Algorithm 3 Scripted Pick and Place

```
pickPos ← target object position
dropPos ← target container position
distThresh ← 0.02
numTimesteps ← 30
placeAttempted ← False
for t in [0, numTimesteps) do
    eePos ← end effector position
    dropPosDist ← ‖eePos − dropPos‖_2
    pickPosDist ← ‖eePos − pickPos‖_2
    if placeAttempted then
        action ← 0
    else if object not grasped AND pickPosDist > distThresh then
        // Move toward target object
        action ← pickPos − eePos
    else if object not grasped then
        // gripper is very close to object
        action ← pickPos − eePos
        close gripper  // Object is in gripper
    else if object not lifted then
        // Move gripper upward to avoid hitting other objects/containers
        action ← [0, 0, 1]
    else if dropPosDist > distThresh then
        // Move toward target container
        action ← dropPos − eePos
    else
        open gripper  // Object falls into container
        placeAttempted ← True
    end if
    noise ∼ 𝒩(0, 0.1)
    action ← action + noise
    s′ ← env.step(action)
end for
```

A.3 Human Language for Our Benchmark

We detail our data collection process for human paraphrases of language instructions.

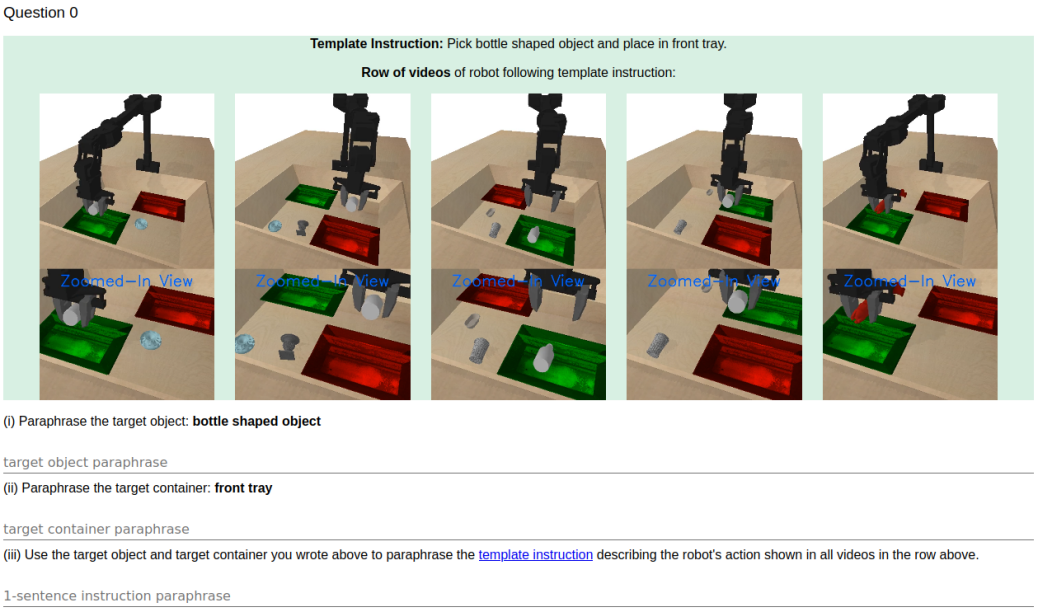


Figure A.1: **Sample Mturk Question.** In each question, we provide the template instruction (described in Appendix 3.5.2.2), a row of 5 robot videos of the task being accomplished, and 3 subquestions asking the respondent to paraphrase: (i) the template target object phrase, (ii) the template target container phrase, and (iii) the entire template instruction with the strings used in (i) and (ii).

A.3.1 HIT layout

For each task, respondents answer 5 questions. A sample question is provided in Figure A.1.

A.3.2 HIT In-Task Quality-Control

Before respondents can work on any questions, they must pass a pre-check consisting of 4 multi-select questions by getting all of the questions correct within 10 tries. These questions mainly acquaint the respondents with how the containers are referred to either by their color or direction, and how objects are referred to either by their name, color, or shape.

To submit the HIT, respondents must provide answers that conform to the following constraints:

- The target object paraphrase must:
  - Be ≥ 5 alphabetical characters long
  - Not contain a set of blocked phrases that are too vague, including "thing" or "target object."
  - Not be a word subsequence of the provided template target object string. (Our subsequence similarity detector still matches subsequences that are a small edit distance from the provided template.)
- The target container paraphrase must:
  - Contain ≥ 2 unique words and be ≥ 6 alphabetical characters long
  - Not contain a set of blocked phrases, including the wrong container identifiers, or the phrases "thing" or "target container."
  - Not be a word subsequence of the provided template target object string.
  - Not be the same as the target object paraphrase
- The instruction paraphrase must:
  - Contain the respondent-inputted target object paraphrase and target container paraphrase
  - Be sufficiently different from the other instruction paraphrases provided by the respondent, as well as the template instruction for this question. (We compare subsequences after removing the target object and target container from each instruction paraphrase.)
  - Be ≥ 15 alphabetical characters long
  - Contain ≥ 2 unique words besides the target object paraphrase and target container paraphrase.
  - Not contain a set of blocked phrases that make the instruction from the wrong point of view, such as "the robot" or "the machine."

To maintain a wide distribution of paraphrases and human noise in the dataset, we do not correct for spelling or grammar errors.

A.3.3 Overall Statistics

We offered a reward of \$0.60 per HIT, allowing a generous 60 mins per HIT. To qualify to work on our HITs, we required respondents to have $\geq$ 10k approved HITs with an approval rate of $\geq 95\%$ to do our HIT. In total, including several initial pilot rounds of mturk data collection (which yielded instructions that were too vague, mistaken, or imprecise to be useful and allowed us to iterate on our HIT design), we spent \$283.68.

Appendix B: Lang4Sim2Real Details

B.1 Scripted Policies for Real-World Data Collection

The following algorithm boxes show the scripted policy functions used to collect demonstrations for each of our three task suites.

Algorithm 4 Scripted Wrap Wire

centerPos ← blender center position
placeAttempted ← False
targetDistToCenter ← 0.15
numTimesteps ← 45
direction ← true if clockwise, false if counterclockwise
for t **in** [0, numTimesteps) **do**
    wirePos ← position of the graspable part of the wire
    eePos ← end effector position
    pickPosDist ← $\|\text{eePos} - \text{wirePos}\|_2$
    done ← is wrapped $> \frac{11\pi}{6}$ from the start to end of wire around centerPos in direction
    if placeAttempted **then**
        action ← 0
    else if object not grasped **AND** pickPosDist > distThresh **then**
        // Move toward wire
        action ← wirePos − eePos
    else if object not grasped **then**
        // gripper is very close to wire
        action ← pickPos − eePos
        close gripper // Object is in gripper
    else if wire not lifted **then**
        action ← [0, 0, 1] // Move up
    else if not done **then**
        relPos ← eePos − centerPos
        distToCenter ← $\|\text{relPos}\|_2$
        normRelPos ← (relPos/distToCenter) ∗ targetDistToCenter
        actionMaintainDistance ← relPos ∗ (targetDistToCenter − distToCenter) // move toward/away from center
        actionMoveTangent ← $[-\text{normRelPos}[1], \text{normRelPos}[0], 0.0]$ // Move tangent to the blender
        if direction **then**
            actionMoveTangent ← actionMoveTangent ∗ −1
        end if
        action ← actionMaintainDistance + actionMoveTangent
    else
        action ← open gripper // Drop wire
        placeAttempted ← True
    end if
end for

Algorithm 5 Scripted Pick and Place Function

Function PickPlace(pickPos, dropPos, distThresh, placeAttempted)
eePos ← end effector position
dropPosDist ← $\|\text{eePos} - \text{dropPos}\|_2$
pickPosDist ← $\|\text{eePos} - \text{pickPos}\|_2$
if placeAttempted **then**
  action ← 0
else if object not grasped **AND** pickPosDist > distThresh **then**
  // Move toward target object
  action ← pickPos − eePos
else if object not grasped **then**
  // gripper is very close to object
  action ← (pickPos − eePos, close gripper) // Object is in gripper
else if object not lifted **then**
  // Move gripper upward to avoid hitting other objects/containers
  action ← [0, 0, 1]
else if dropPosDist > distThresh **then**
  // Move toward target container
  action ← dropPos − eePos
else
  action ← open gripper // Object falls into container
  placeAttempted ← True
end if
noise $\sim \mathcal{N}(0, 0.1)$
action ← action + noise
return action, placeAttempted
End Function

Algorithm 6 Stack Object

```
pickPos ← target object position
dropPos ← target container position
numTimesteps ← 18
distThresh ← 0.02
placeAttempted ← False
for t in [0, numTimesteps) do
    action, placeAttempted ← PickPlace(pickPos, dropPos, distThresh, placeAttempted)
    s' ← env.step(action)
end for
```

Algorithm 7 Scripted 2-step Pick and Place

pickPos ← [object position, first container position]
dropPos ← [first container position, second container position]
numTimesteps ← 45
distThresh ← 0.02
placeAttempted ← [False, False]
$s_i \leftarrow 0$ // step index (starts at 0, and increments to 1 when first pick-place step is complete)
stepCompleted ← [False, False]
for t **in** [0, numTimesteps) **do**
    action, placeAttempted[s_i] ← PickPlace(pickPos[s_i], dropPos[s_i], distThresh, placeAttempted[s_i])
    if stepIsSuccessful(s_i) **AND not** stepsCompleted[s_i] **then**
        stepsCompleted[s_i] ← True
        $s_i \leftarrow 1$
    end if
    $s' \leftarrow$ env.step(action)
end for

B.2 Language Granularity Experiments

Table B.1: `sim2real`: Language annotations and language granularity on 2-step real-world pick-and-place

All-stages	Half-stages	2-stage	1-stage
gripper open, reaching for carrot, out of bowl	gripper open, reaching for carrot, out of bowl	picking carrot and putting in bowl	random language embedding
gripper open, moving down over carrot, out of bowl			
gripper closing, with carrot, out of bowl	gripper closing, with carrot, out of bowl		
gripper closed, moving up with carrot, out of bowl	gripper closed, moving up with carrot		
gripper closed, moving sideways with carrot, out of bowl			
gripper closed, with carrot, above bowl			
gripper open, dropped carrot, in bowl	gripper open, dropped carrot, in bowl		
gripper open, reaching for bowl, out of plate	gripper open, reaching for bowl, out of plate	picking bowl and putting in plate	
gripper open, moving down over bowl, out of plate			
gripper closing, with bowl, out of bowl	gripper closing, with bowl, out of plate		
gripper closed, moving up with bowl, out of plate	gripper closed, moving up with bowl		
gripper closed, moving sideways with carrot, out of bowl			
gripper closed, with bowl, above plate			
gripper open, dropped bowl, in plate	gripper open, dropped bowl, in plate		

Table B.2: `sim2real`: Language annotations and language granularity on wire wrap

All-stages	half-stages	2-stage	1-stage
gripper open, reaching for plug	gripper open, reaching for plug	picking and wrapping beads around cylinder	random language embedding
gripper open, moving down over plug			
gripper closing around plug	gripper closing and lifting plug		
gripper closed, moving up with plug			
counter-clockwise left	counter-clockwise		
counter-clockwise front			
counter-clockwise right			
counter-clockwise back			
clockwise left	clockwise		
clockwise front			
clockwise right			
clockwise back			
gripper open, above plug with wire fully wrapped	gripper open, above blender with wire fully wrapped	beads fully wrapped	
gripper open, above plug with wire fully unwrapped	gripper open, above blender with wire fully unwrapped		

Appendix C: MICoBot Details

C.1 Detailed Simulation Results

Table C.1: Simulation Task 1 Performance across different $\tilde{p}_{H,t}$ Values and Language Sentiments.

Method	Metric	Human Parameters (Mood, $\tilde{p}_{H,t}$)								Avg. (%)
		Positive Mood				Negative Mood				
		0.0	**0.3**	**0.7**	**1.0**	**0.0**	**0.3**	**0.7**	**1.0**	
Ours	Success Rate	3/10	6/10	9/10	10/10	1/10	4/10	9/10	9/10	**63.75**
	Num Plan Steps Completed	3.6/5	4.2/5	4.8/5	5.0/5	3.2/5	3.8/5	4.8/5	4.5/5	**84.5**
	Prop. Plan Steps done by Human	0.1667	0.2381	0.3125	0.4	0.03125	0.1579	0.354	0.377	25.47
LLM Baseline	Success Rate	2/10	2/10	4/10	7/10	3/10	6/10	6/10	6/10	45
	Num Plan Steps Completed	3.4/5	3.4/5	3.7/5	4.4/5	3.6/5	4.2/5	4.0/5	4.2/5	77.25
	Prop. Plan Steps done by Human	0.0588	0.05882	0.2162	0.1591	0.1111	0.1428	0.175	0.166	13.6
Random Agent	Success Rate	2/10	5/10	6/10	7/10	2/10	3/10	6/10	7/10	47.5
	Num Plan Steps Completed	3.4/5	3.5/5	4.0/5	4.4/5	3.4/5	2.8/5	4.0/5	4.4/5	74.75
	Prop. Plan Steps done by Human	0.1176	0.4286	0.525	0.7045	0.1176	0.2143	0.525	0.7045	41.71
RL	Success Rate	0/10	1/10	4/10	10/10	0/10	1/10	4/10	10/10	37.5
	Num Plan Steps Completed	2.4/5	2.3/5	3.4/5	5.0/5	2.4/5	2.3/5	3.4/5	5.0/5	65.5
	Prop. Plan Steps done by Human	0.125	0.1739	0.4412	0.54	0.125	0.1739	0.4412	0.54	32.0
Only R Init	Success Rate	0/10	3/10	9/10	10/10	0/10	5/10	9/10	10/10	57.5
	Num Plan Steps Completed	3.0/5	3.6/5	4.8/5	5.0/5	3.0/5	4.0/5	4.8/5	5.0/5	83
	Prop. Plan Steps done by Human	0.0	0.1111	0.3542	0.4	0.0	0.225	0.354	0.4	23.05
Only H Init	Success Rate	0/10	0/10	0/10	0/10	2/10	0/10	0/10	2/10	5.0
	Num Plan Steps Completed	3.0/5	3.0/5	3.0/5	3.0/5	3.2/5	3.0/5	3.0/5	3.3/5	61.25
	Prop. Plan Steps done by Human	0.0	0.0	0.0	0.0/3.0	0.1875	0.0	0.0	0.1212	3.86
Ours w/o p_help	Success Rate	3/10	5/10	9/10	10/10	2/10	3/10	9/10	9/10	62.5
	Num Plan Steps Completed	3.6/5	4.0/5	4.8/5	5.0/5	3.4/5	3.4/5	4.8/5	4.7/5	84.25
	Prop. Plan Steps done by Human	0.1667	0.3	0.3333	0.38	0.1176	0.2059	0.3125	0.4468	28.29
Ours w/o Plan Hier.	Success Rate	2/10	4/10	7/10	10/10	0/10	3/10	4/10	8/10	47.5
	Num Plan Steps Completed	3.4/5	3.8/5	4.0/5	5.0/5	3.0/5	3.4/5	3.6/5	4.2/5	76
	Prop. Plan Steps done by Human	0.0588	0.1316	0.25	0.24	0.0667	0.1176	0.1944	0.2381	16.22

C.2 User Study Instructions

Users were read the following instructions at the beginning of the study. (Instructions here are shown for Task 2.)

1. Thank you so much for coming for our user study! We wanted to remind you to review the RIS before proceeding, and that you may voluntarily opt-out of the study at any time.

2. You are working with the robot to perform the task of assembling the toy car. You must use the hexagonal drill bit to screw in the wheels, and the phillips

drill bit to screw in the seat and the window. [Demonstrate these steps to the human]. You and the robot operate on a shared understanding of the plan. [Read the 4 high-level steps of the plan tree for this task. Do NOT discuss the low-level steps of the plan tree.]

3. Our goal is to simulate a home robot setting, where the human (you) is relatively busy with your own tasks, and once in a while you provide physical assistance and talk to the robot. So you are free to do work during each trial.

4. Once the robot asks you to do a step, and you accept, you must finish that step successfully.

5. We will perform 2 trials, each of a different method.

6. Both you and the robot can do a subset of the steps in the plan. You will communicate with the robot to determine who does what steps.

7. These are the objects you will work with during the task. I will move them now to their initial positions where they will start at the beginning of each trial. [Move objects to initial positions.]

8. For safety, I will gate-keep each of the robot's physical actions. In other words, the actions are generated by the robot itself, but they will be displayed on the laptop screen with a confirmation message, and I can either allow that physical action to be executed by the robot, or block the action from being executed if it brings the robot to an unsafe location.

9. The robot will stay on the TV side of the coffee table, while you will sit on the couch and stay on the couch side of the coffee table.

10. You are free to get up off the couch if you want to volunteer to perform steps that involve going to the sink or shelf, but you can only go when the robot is stationary and waiting on the other side of the coffee table. Steps are done

in sequential order; our system doesn't support parallelization (agents working simultaneously).

11. You will be communicating to the robot through this headset. We will perform a mic-check now to make sure it can pick up your voice. [Do mic check.]

12. Now, this is what the robot will sound like when it talkes to you. [Play audio sample of the robot.] Try responding to it, and I will see if it can hear you.

13. The systems today can handle different kinds of dialog. (1) refusal/acceptance, (2) task allocation, such as ("Could you pour the package in the plate later?" Or: "I can pour the package onto the plate later."), (3) silence—you don't need to respond to the robot every time, and (4) a proposal to split up adjacent steps, such as "Please bring me the drill so that I can put on the wheels." You may engage in any of these types of dialog, and the robot may also engage in them when communicating to you.

14. Do you have any questions before we start? I will let you know when each trial begins and ends. Sometimes trials may end prematurely.

C.3 Real-world Failure Analysis

See the failure breakdowns in our real-world trials for MICoBot in Figure 5.10.

1. Task 1: Cut and Pour Package into Bowl. 0 failed trials out of 6.

2. Task 2: Assemble Toy Car. 2 failed trials out of 6. 1 failure that was rectified by human.

    - 1 dialog error: user wanted robot to perform a non-step plan outside of its capabilities, and refused when robot said it wasn't able to perform it
    - 1 primitive error: robot did not release its grasp of the drill.

- 1 metaplanner dialog parsing error

3. Task 3: Pack Gift Box. 2 failed trials out of 6.
    - 1 primitive error: placed bow on box but bow dropped to floor as gripper retracted
    - 1 termination condition triggered: human rejected robot's help request and subsequent proposals twice in a row.

See the failure breakdowns in our real-world trials for the LLM baseline in Figure 5.11.

1. Task 1: Cut and Pour Package into Bowl. 1 failed trials out of 6.
    - 1 termination condition triggered: human rejected robot's help request twice in a row.
2. Task 2: Assemble Toy Car. 6 failed trials out of 6.
    - 2 primitive errors: Placed the drill, but then it fell off of the table.
    - 2 perception errors: Unable to visually find the correct location to place the object.
    - 1 task allocation error: Allocated to put on the car wheels itself.
    - 1 termination condition triggered: Fell into a conversational loop with the user.
3. Task 3: Pack Gift Box. 6 failed trials out of 6. 1 failure was rectified by human.
    - 6 task allocation errors: Tried to put on the lid itself four times; tried to cut a piece of tape itself twice.
    - 1 primitive error: Inadvertently dropped the car onto the floor as it was trying to place it.

Works Cited

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. Vqa: Visual question answering. *International Journal of Computer Vision*, 123:4 – 31, 2015. URL `https://api.semanticscholar.org/CorpusID:3180429`.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as I can and not as I say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

Bo Ai, Zhanxin Wu, and David Hsu. Invariance is key to generalization: Examining the role of representation in sim-to-real transfer for visual navigation. *arXiv preprint arXiv:2310.15020*, 2023.

Carl Allen and Timothy Hospedales. Analogies explained: Towards understanding word embeddings, 2019. URL `https://arxiv.org/abs/1901.09813`.

J.E. Allen, C.I. Guinn, and E. Horvtz. Mixed-initiative interaction. *IEEE Intelligent Systems and their Applications*, 14(5):14–23, 1999. doi: 10.1109/5254.796083.

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning (ICML)*, 2017.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Montserrat Gonzalez Arenas, Ted Xiao, Sumeet Singh, Vidhi Jain, Allen Ren, Quan Vuong, Jake Varley, Alexander Herzog, Isabel Leal, Sean Kirmani, et al. How to prompt your robot: A promptbook for manipulation skills with code as policies. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4340–4348. IEEE, 2024.

Brenna D. Argall and Aude G. Billard. A survey of tactile human–robot interactions. *Robotics and Autonomous Systems*, 58(10):1159–1176, 2010. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2010.07.002. URL `https://www.sciencedirect.com/science/article/pii/S0921889010001375`.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34:26–38, 2017. URL `https://api.semanticscholar.org/CorpusID:4884302`.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022a.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, John Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Chris Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, E Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, J Landau, Kamal Ndousse, Kamile Lukosiute, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noem'i Mercado, Nova Dassarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, T. J. Henighan, Tristan Hume, Sam Bowman, Zac Hatfield-Dodds, Benjamin Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom B. Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback. *ArXiv*, abs/2212.08073, 2022b. URL `https://api.semanticscholar.org/CorpusID:254823489`.

Jimmy Baraglia, Maya Cakmak, Yukie Nagai, Rajesh Rao, and Minoru Asada. Initiative in robot assistance during collaborative task execution. In *HRI*, pages 67–74, 2016. doi: 10.1109/HRI.2016.7451735.

Samuel Barrett, Noa Agmon, Noam Hazon, Sarit Kraus, and Peter Stone. Communicating with unknown teammates. In *Proceedings of the Twenty-First European Conference on Artificial Intelligence*, August 2014.

Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. Rt-h: Action hierarchies using language, 2024. URL `https://arxiv.org/abs/2403.01823`.

Adrien Bennetot, Vicky Charisi, and Natalia Díaz-Rodríguez. Should artificial agents ask for help in human-robot collaborative problem-solving?, 2020. URL `https://arxiv.org/abs/2006.00882`.

Justin Bishop, Jaylen Burgess, Cooper Ramos, Jade B. Driggs, Tom Williams, Chad C. Tossell, Elizabeth Phillips, Tyler H. Shaw, and Ewart J. de Visser. Chaopt: A testbed for evaluating human-autonomy team collaboration using the video game overcooked!2. In *2020 Systems and Information Engineering Design Symposium (SIEDS)*, pages 1–6, 2020. doi: 10.1109/SIEDS49339.2020.9106686.

Valts Blukis, Dipendra Misra, Ross A. Knepper, and Yoav Artzi. Mapping navigation instructions to continuous control actions with position-visitation prediction. In *Conference on Robot Learning (CoRL)*, 2018.

Valts Blukis, Yannick Terme, Eyvind Niklasson, Ross A. Knepper, and Yoav Artzi. Learning to map natural language instructions to physical quadcopter control using simulated flight. In *Conference on Robot Learning (CoRL)*, 2019.

Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731, 2017.

SRK Branavan, David Silver, and Regina Barzilay. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704, 2012.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020.

Kalesha Bullard, Andrea L Thomaz, and Sonia Chernova. Towards intelligent arbitration of diverse active learning queries. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6049–6056. IEEE, 2018.

Jaime R. Carbonell. Ai in cai: An artificial-intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, 11(4):190–202, 1970. doi: 10.1109/TMMS.1970.299942.

Silvia Cascianelli, Gabriele Costante, Thomas A Ciarfuglia, Paolo Valigi, and Mario L Fravolini. Full-gru natural language video description for service robotics applications. *IEEE RA-L*, 3(2):841–848, 2018.

Jun Cen, Siteng Huang, Yuqian Yuan, Kehan Li, Hangjie Yuan, Chaohui Yu, Yuming Jiang, Jiayan Guo, Xin Li, Hao Luo, et al. Rynnvla-002: A unified vision-language-action and world model. *arXiv preprint arXiv:2511.17502*, 2025.

Andrea Censi, Konstantin Slutsky, Tichakorn Wongpiromsarn, Dmitry Yershov, Scott Pendleton, James Fu, and Emilio Frazzoli. Liability, ethics, and culture-aware behavior specification using rulebooks. In *2019 international conference on robotics and automation (ICRA)*, pages 8536–8542. IEEE, 2019a.

Andrea Censi, Konstantin Slutsky, Tichakorn Wongpiromsarn, Dmitry S. Yershov, Scott Pendleton, James Guo Ming Fu, and Emilio Frazzoli. Liability, ethics, and culture-aware behavior specification using rulebooks. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8536–8542, 2019b. URL `https://api.semanticscholar.org/CorpusID:67856233`.

Boyuan Chen, Fei Xia, Brian Ichter, Kanishka Rao, Keerthana Gopalakrishnan, Michael S. Ryoo, Austin Stone, and Daniel Kappler. Open-vocabulary queryable scene representations for real world planning. In *arXiv preprint arXiv:2209.09874*, 2022.

David L. Chen, Joohyun Kim, and Raymond J. Mooney. Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research*, 37:397–435, 2010. URL `http://www.cs.utexas.edu/users/ai-lab?chen:jair10`.

Delong Chen, Theo Moutakanni, Willy Chung, Yejin Bang, Ziwei Ji, Allen Bolourchi, and Pascale Fung. Planning with reasoning using vision language world model. *arXiv preprint arXiv:2509.02722*, 2025.

Hao Chen, Ran Tao, Han Zhang, Yidong Wang, Xiang Li, Wei Ye, Jindong Wang, Guosheng Hu, and Marios Savvides. Conv-adapter: Exploring parameter efficient transfer learning for convnets, 2024a.

Lawrence Yunliang Chen, Kush Hari, Karthik Dharmarajan, Chenfeng Xu, Quan Vuong, and Ken Goldberg. Mirage: Cross-embodiment zero-shot policy transfer with cross-painting. *ArXiv*, abs/2402.19249, 2024b. URL `https://api.semanticscholar.org/CorpusID:268063346`.

Maximillian Chen, Xiao Yu, Weiyan Shi, Urvi Awasthi, and Zhou Yu. Controllable mixed-initiative dialogue generation through prompting. *arXiv preprint arXiv:2305.04147*, 2023.

Maximillian Chen, Ruoxi Sun, Sercan O. Arık, and Tomas Pfister. Learning to clarify: Multi-turn conversations with action-based contrastive self-training, 2024c. URL `https://arxiv.org/abs/2406.00222`.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Universal image-text representation

learning. In *European Conference on Computer Vision*, 2019. URL https://api.semanticscholar.org/CorpusID:216080982.

Jae-Woo Choi, Hyungmin Kim, Hyobin Ong, Youngwoo Yoon, Minsu Jang, Jaehong Kim, et al. Reactree: Hierarchical task planning with dynamic tree expansion using llm agent nodes. 2025.

Jennifer Chu-Carroll. MIMIC: An adaptive mixed initiative spoken dialogue system for information queries. In *Sixth Applied NLP Conference*, pages 97–104, Seattle, Washington, USA, April 2000. ACL. doi: 10.3115/974147.974161. URL https://aclanthology.org/A00-1014/.

Erwin Coumans and Yunfei Bai. Bullet and pybullet, physics simulation for games, visual effects, robotics and reinforcement learning. https://pybullet.org, May 2007-2022.

Yuchen Cui, Scott Niekum, Abhinav Gupta, Vikash Kumar, and Aravind Rajeswaran. Can foundation models perform zero-shot task specification for robot manipulation? *Learning for Dynamics and Control Conference*, 2022.

Longchao Da, Justin Turnau, Thirulogasankar Pranav Kutralingam, Alvaro Velasquez, Paulo Shakarian, and Hua Wei. A survey of sim-to-real methods in rl: Progress, prospects and challenges with foundation models. *ArXiv*, abs/2502.13187, 2025. URL https://api.semanticscholar.org/CorpusID:276449804.

Nicola Dainese, Pekka Marttinen, and Alexander Ilin. Reader: Model-based language-instructed reinforcement learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16583–16599, 2023.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett,

Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.

Abhishek Das, Satwik Kottur, Jose M. F. Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: a speech-text foundation model for real-time dialogue. Technical report, 2024. URL `https://arxiv.org/abs/2410.00037`.

Robin Deits, Stefanie Tellex, Pratiksha Thaker, Dimitar Simeonov, Thomas Kollar, and Nicholas Roy. Clarifying commands with information-theoretic human-robot dialog. *J. Hum.-Robot Interact.*, 2(2):58–79, June 2013. doi: 10.5898/JHRI.2.2.Deits. URL `https://doi.org/10.5898/JHRI.2.2.Deits`.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Yang Deng, Wenqiang Lei, Wai Lam, and Tat-Seng Chua. A survey on proactive dialogue systems: Problems, methods, and prospects. *arXiv preprint arXiv:2305.02750*, 2023a.

Yang Deng, Lizi Liao, Liang Chen, Hongru Wang, Wenqiang Lei, and Tat-Seng Chua. Prompting and evaluating large language models for proactive dialogues: Clarification, target-guided, and non-collaboration, 2023b. URL `https://arxiv.org/abs/2305.13626`.

Yang Deng, Wenxuan Zhang, Wai Lam, See-Kiong Ng, and Tat-Seng Chua. Plug-and-play policy planner for large language model powered dialogue agents, 2024. URL https://arxiv.org/abs/2311.00262.

Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *International Conference on Robotics and Automation (ICRA)*, 2017.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation, 2015. URL https://arxiv.org/abs/1410.8516.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp, 2017. URL https://arxiv.org/abs/1605.08803.

Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

Clemens Eppner, Arsalan Mousavian, and Dieter Fox. Acronym: A large-scale grasp dataset based on simulation, 2020. URL https://arxiv.org/abs/2011.09584.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pages 1407–1416. PMLR, 2018.

Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. Towards understanding linear word analogies, 2019. URL https://arxiv.org/abs/1810.04882.

Maurizio Faccio, Irene Granata, and Riccardo Minto. Task allocation model for human-robot collaboration with variable cobot speed. *Journal of Intelligent Manufacturing*, 35:793–806, 2024. doi: 10.1007/s10845-023-02073-9. URL https://link.springer.com/article/10.1007/s10845-023-02073-9.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362, 2022.

Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 11444–11453, 2020.

Hao-Shu Fang, Minghao Gou, Chenxi Wang, and Cewu Lu. Robust grasping across diverse sensor qualities: The graspnet-1billion dataset. *The International Journal of Robotics Research*, 2023.

Xueyang Feng, Zhi-Yuan Chen, Yujia Qin, Yankai Lin, Xu Chen, Zhiyuan Liu, and Ji-Rong Wen. Large language model-based human-agent collaboration for complex task solving, 2024. URL https://arxiv.org/abs/2402.12914.

Douglas A. Few, David J. Bruemmer, and Miles C. Walton. Improved human-robot teaming through facilitated initiative. In *ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 171–176, 2006. doi: 10.1109/ROMAN.2006.314413.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *International Conference on Robotics and Automation (ICRA)*, 2016.

Qiaozi Gao, Govind Thattai, Suhaila Shakiah, Xiaofeng Gao, Shreyas Pansare, Vasu Sharma, Gaurav Sukhatme, Hangjie Shi, Bofei Yang, Desheng Zheng, Lucy Hu, Karthika Arumugam, Shui Hu, Matthew Wen, Dinakar Guthy, Cadence Chung, Rohan Khanna, Osman Ipek, Leslie Ball, Kate Bland, Heather Rocker, Yadunandana Rao, Michael Johnston, Reza Ghanadan, Arindam Mandal, Dilek Hakkani Tur, and Prem Natarajan. Alexa arena: A user-centric interactive platform for embodied ai, 2023. URL `https://arxiv.org/abs/2303.01586`.

Divyansh Garg, Skanda Vaidyanath, Kuno Kim, Jiaming Song, and Stefano Ermon. Lisa: Learning interpretable skill abstractions from language, 2022. URL `https://arxiv.org/abs/2203.00054`.

Majid Ghasemi, Amir Hossein Moosavi, and Dariush Ebrahimi. A comprehensive survey of reinforcement learning: From algorithms to practical challenges. 2024. URL `https://api.semanticscholar.org/CorpusID:274422589`.

Muhammad Fadhil Ginting, Dong-Ki Kim, Sung-Kyun Kim, Bandi Jai Krishna, Mykel J Kochenderfer, Shayegan Omidshafiei, and Ali-akbar Agha-mohammadi. Saycomply: Grounding field robotic tasks in operational compliance through retrieval-based language models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13730–13736. IEEE, 2025.

John Gkountouras, Matthias Lindemann, Phillip Lippe, Efstratios Gavves, and Ivan Titov. Language agents meet causality–bridging llms and causal world models. In *International Conference on Learning Representations*, volume 2025, pages 37311–37336, 2025.

Prasoon Goyal, Scott Niekum, and Raymond Mooney. Using natural language for reward shaping in reinforcement learning. 2019. URL `https://arxiv.org/abs/1903.02020`.

Prasoon Goyal, Scott Niekum, and Raymond Mooney. Pixl2r: Guiding reinforcement learning using natural language by mapping pixels to rewards. 2020. URL https://arxiv.org/abs/2007.15543.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.

Jennifer Grannen, Siddharth Karamcheti, Suvir Mirchandani, Percy Liang, and Dorsa Sadigh. Vocal sandbox: Continual learning and adaptation for situated human-robot collaboration. *arXiv preprint arXiv:2411.02599*, 2024.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, et al. Ego4d: Around the world in 3,000 hours of egocentric video. *arXiv preprint arXiv:2110.07058*, 2021.

Weiwei Gu, Suresh Kondepudi, Anmol Gupta, Lixiao Huang, and Nakul Gopalan. Continual robot skill and task learning via dialogue, 2026. URL https://arxiv.org/abs/2409.03166.

Christian Gumbsch, Noor Sajid, Georg Martius, and Martin V. Butz. Learning hierarchical world models with adaptive temporal abstractions from discrete latent dynamics. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=TjCDNssXKU.

Abhishek Gupta, Corey Lynch, Brandon Kinman, Garrett Peake, Sergey Levine, and Karol Hausman. Demonstration-bootstrapped autonomous practicing via multi-task reinforcement learning. *arXiv preprint arXiv:2203.15755*, 2022a.

Agrim Gupta, Stephen Tian, Yunzhi Zhang, Jiajun Wu, Roberto Martín-Martín, and Li Fei-Fei. Maskvit: Masked visual pre-training for video prediction. *arXiv preprint arXiv:2206.11894*, 2022b.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pages 2451–2463. Curran Associates, Inc., 2018. URL `https://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution`. `https://worldmodels.github.io`.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations (ICLR)*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

Austin W Hanjie, Victor Y Zhong, and Karthik Narasimhan. Grounding language to entities and dynamics for generalization in reinforcement learning. In *International conference on machine learning*, pages 4051–4062. PMLR, 2021.

Nicklas Hansen and Xiaolong Wang. Hallucination in world models is predictable and preventable, 2026.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3796–3803, 2019.

Daniel Ho, Kanishka Rao, Zhuo Xu, Eric Jang, Mohi Khansari, and Yunfei Bai. Retinagan: An object-aware approach to sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10920–10926. IEEE, 2021.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL `https://arxiv.org/abs/2106.09685`.

Jiaheng Hu, Jay Shim, Chen Tang, Yoonchang Sung, Bo Liu, Peter Stone, and Roberto Martin-Martin. Simple recipe works: Vision-language-action models are natural continual learners with reinforcement learning. *arXiv preprint arXiv:2603.11653*, 2026.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *arXiv preprint arXiv:2207.05608*, 2022.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models, 2023. URL `https://arxiv.org/abs/2307.05973`.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. In *ACM Computing Surveys*, 2017.

Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-z: Zero-shot task generalization with robotic imitation learning. In *5th Annual Conference on Robot Learning*, 2021. URL `https://openreview.net/forum?id=8kbp23tSGYv`.

Shu Jiang and Ronald C. Arkin. Mixed-initiative human-robot interaction: Definition, taxonomy, and survey. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 954–961, 2015. doi: 10.1109/SMC.2015.174.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv*, 2022.

Emily Jin, Jiaheng Hu, Zhuoyi Huang, Ruohan Zhang, Jiajun Wu, Li Fei-Fei, and Roberto Martín-Martín. Mini-behavior: A procedurally generated benchmark for long-horizon decision-making in embodied ai. *arXiv preprint 2310.01824*, 2023.

Ya Jing, Xuelin Zhu, Xingbin Liu, Qie Sima, Taozheng Yang, Yunhai Feng, and Tao Kong. Exploring visual pre-training for robot manipulation: Datasets, models and methods. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11390–11395. IEEE, 2023.

Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

Siddharth Karamcheti, Megha Srivastava, Percy Liang, and Dorsa Sadigh. Lila: Language-informed latent actions. In *5th Annual Conference on Robot Learning*, 2021. URL `https://arxiv.org/pdf/2111.03205`.

Manuel Kaspar, Juan D Muñoz Osorio, and Jürgen Bock. Sim2real transfer for reinforcement learning without dynamics randomization. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4383–4388. IEEE, 2020.

Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le,

Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Multimodal neural language models. In *International Conference on Machine Learning*, 2014. URL `https://api.semanticscholar.org/CorpusID:12365096`.

Ross A. Knepper, Todd Layton, John Romanishin, and Daniela Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *2013 IEEE International Conference on Robotics and Automation*, pages 855–862, 2013. doi: 10.1109/ICRA.2013.6630673.

Noriyuki Kojima, Alane Suhr, and Yoav Artzi. Continual learning for grounded instruction generation by observing human following behavior, 2021. URL `https://arxiv.org/abs/2108.04812`.

Benjamin Kolb, Leon Lang, Henning Bartsch, Arwin Gansekoele, Raymond Koopmanschap, Leonardo Romor, David Speck, Mathijs Mul, and Elia Bruni.

Learning to request guidance in emergent communication, 2019. URL https://arxiv.org/abs/1912.05525.

Satwik Kottur, José Moura, Stefan Lee, and Dhruv Batra. Natural language does not emerge 'naturally' in multi-agent dialog. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2962–2967, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1321. URL https://aclanthology.org/D17-1321/.

Johannes Kulick, Marc Toussaint, Tobias Lang, and Manuel Lopes. Active learning for teaching a robot grounded relational symbols. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, page 1451–1457. AAAI Press, 2013. ISBN 9781577356332.

Thinking Machines Lab. Interaction models: A scalable approach to human-ai collaboration. *Thinking Machines Lab: Connectionism*, May 2026. doi: 10.64434/tml.20260511. https://thinkingmachines.ai/blog/interaction-models/.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

Omer Levy and Yoav Goldberg. Linguistic regularities in sparse and explicit word representations. In Roser Morante and Scott Wen-tau Yih, editors, *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180, Ann Arbor, Michigan, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-1618. URL https://aclanthology.org/W14-1618/.

Amber Li and Tom Silver. Embodied active learning of relational state abstractions for bilevel planning. *arXiv preprint arXiv:2303.04912*, 2023.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Silvio Savarese, Hyowon Gweon, Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1k: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *CoRL*, 2022. URL https://openreview.net/forum?id=_8DoIe8G3t.

Chengshu Li, Jacky Liang, Andy Zeng, Xinyun Chen, Karol Hausman, Dorsa Sadigh, Sergey Levine, Li Fei-Fei, Fei Xia, and Brian Ichter. Chain of code: Reasoning with a language model-augmented code emulator. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 28259–28277. PMLR, 21–27 Jul 2024a. URL https://proceedings.mlr.press/v235/li24ar.html.

Chengshu Li, Jacky Liang, Andy Zeng, Xinyun Chen, Karol Hausman, Dorsa Sadigh, Sergey Levine, Li Fei-Fei, Fei Xia, and Brian Ichter. Chain of code: Reasoning with a language model-augmented code emulator. volume 235. ICML, 2024b. URL https://proceedings.mlr.press/v235/li24ar.html.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *ArXiv*, abs/1908.03557, 2019. URL https://api.semanticscholar.org/CorpusID:199528533.

Xinhai Li, Jialin Li, Ziheng Zhang, Rui Zhang, Fan Jia, Tiancai Wang, Haoqiang Fan, Kuo-Kun Tseng, and Ruiping Wang. Robogsim: A real2sim2real

robotic gaussian splatting simulator. *ArXiv*, abs/2411.11839, 2024c. URL https://api.semanticscholar.org/CorpusID:274130540.

Yitong Li, Martin Renqiang Min, Dinghan Shen, David Edwin Carlson, and Lawrence Carin. Video generation from text. In *AAAI Conference on Artificial Intelligence*, 2017. URL https://api.semanticscholar.org/CorpusID:8672818.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*, 2022.

Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca Dragan. Learning to model the world with language. *arXiv preprint arXiv:2308.01399*, 2023.

Toru Lin, Minyoung Huh, Chris Stauffer, Ser-Nam Lim, and Phillip Isola. Learning to ground multi-agent communication with autoencoders, 2021. URL https://arxiv.org/abs/2110.15349.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qian Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *ArXiv*, abs/2306.03310, 2023a. URL https://api.semanticscholar.org/CorpusID:259089508.

Huihan Liu, Changyeon Kim, Bo Liu, Minghuan Liu, and Yuke Zhu. Pre-trained vision-language-action models are surprisingly resistant to forgetting in continual learning. *arXiv preprint arXiv:2603.03818*, 2026.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection, 2023b.

Kenzo Lobos-Tsunekawa, Akshay Srinivasan, and Michael Spranger. Madreamer: Coordination and communication through shared imagination, 2022. URL https://arxiv.org/abs/2204.04687.

Xingzhou Lou, Junge Zhang, Ziyan Wang, Kaiqi Huang, and Yali Du. Safe reinforcement learning with free-form natural language constraints and pretrained language models. *arXiv preprint arXiv:2401.07553*, 2024.

Ryan Lowe, Abhinav Gupta, Jakob Foerster, Douwe Kiela, and Joelle Pineau. On the interaction between supervision and self-play in emergent communication, 2020. URL https://arxiv.org/abs/2002.01093.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Neural Information Processing Systems*, 2019. URL https://api.semanticscholar.org/CorpusID:199453025.

Xusheng Luo, Shaojun Xu, and Changliu Liu. Obtaining hierarchy from human instructions: an llms-based approach. In *CoRL 2023 Workshop on Learning Effective Abstractions for Planning (LEAP)*, 2023.

Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *Robotics: Science and Systems*, 2021. URL https://arxiv.org/abs/2005.07648.

Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. *arXiv preprint arXiv:2306.00958*, 2023.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models, 2024a. URL https://arxiv.org/abs/2310.12931.

Yecheng Jason Ma, William Liang, Hungju Wang, Sam Wang, Yuke Zhu, Linxi Fan, Osbert Bastani, and Dinesh Jayaraman. Dreureka: Language model guided sim-to-real transfer. In *Robotics: Science and Systems (RSS)*, 2024b.

William Macke. Optimizing and planning with queries for communication in ad hoc teamwork. Master's thesis, University of Texas at Austin, 2023. URL https://repositories.lib.utexas.edu/server/api/core/bitstreams/5e958e0a-042e-434 content.

Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1957–1964. IEEE, 2016.

Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. 2017.

Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models, 2023. URL https://arxiv.org/abs/2307.04738.

Elman Mansimov, Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov. Generating images from captions with attention. *CoRR*, abs/1511.02793, 2015. URL https://api.semanticscholar.org/CorpusID:9996719.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan Loddon Yuille. Explain images with multimodal recurrent neural networks. *ArXiv*, abs/1410.1090, 2014. URL https://api.semanticscholar.org/CorpusID:3527896.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3190–3199, 2019. URL `https://api.semanticscholar.org/CorpusID:173991173`.

Jan Matas, Stephen James, and Andrew J Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 734–743. PMLR, 2018.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks, 2021. URL `https://arxiv.org/abs/2112.03227`.

Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned imitation learning. *arXiv preprint arXiv:2204.06252*, 2022.

Yuan Meng, Zhenguo Sun, Max Fest, Xukun Li, Zhenshan Bing, and Alois Knoll. Growing with your embodied agent: A human-in-the-loop lifelong code generation framework for long-horizon manipulation skills. *arXiv preprint arXiv:2509.18597*, 2025.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013. URL `https://arxiv.org/abs/1310.4546`.

Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Yao Mu, Junting Chen, Qinglong Zhang, Shoufa Chen, Qiaojun Yu, Chongjian Ge, Runjian Chen, Zhixuan Liang, Mengkang Hu, Chaofan Tao, Peize Sun,

Haibao Yu, Chao Yang, Wenqi Shao, Wenhai Wang, Jifeng Dai, Yu Qiao, Mingyu Ding, and Ping Luo. Robocodex: Multimodal code generation for robotic behavior synthesis. *ArXiv*, abs/2402.16117, 2024. URL https://api.semanticscholar.org/CorpusID:267938053.

Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun. Driving policy transfer via modularity and abstraction. *arXiv preprint arXiv:1804.09364*, 2018.

Bilge Mutlu, Jodi Forlizzi, and Jessica Hodgins. A storytelling robot: Modeling and evaluation of human-like gaze behavior. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 518–523, 2006. doi: 10.1109/ICHR.2006.321322.

Vivek Myers, Chongyi Zheng, Anca Dragan, Sergey Levine, and Benjamin Eysenbach. Learning temporal distances: Contrastive successor features can provide a metric structure for decision-making, 2025. URL https://arxiv.org/abs/2406.17098.

Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *International Conference on Robotics and Automation (ICRA)*, 2017. URL https://arxiv.org/abs/1703.02018.

Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. URL https://arxiv.org/abs/1807.04742.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets, 2021a. URL https://arxiv.org/abs/2006.09359.

Suraj Nair, Eric Mitchell, Kevin Chen, Brian Ichter, Silvio Savarese, and Chelsea Finn. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *5th Annual Conference on Robot Learning*, 2021b. URL https://arxiv.org/pdf/2109.01115.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning, 2024. URL https://arxiv.org/abs/2303.05479.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. Grounding language for transfer in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 63:849–874, 2018.

Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. 2019.

Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie, Danny Driess, Ayzaan Wahid, Zhuo Xu, Quan Vuong, Tingnan Zhang, Tsang-Wei Edward Lee, Kuang-Huei Lee, Peng Xu, Sean Kirmani, Yuke Zhu, Andy Zeng, Karol Hausman, Nicolas Heess, Chelsea Finn, Sergey Levine, and Brian Ichter. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. 2024.

Manisha Natarajan, Chunyue Xue, Sanne van Waveren, Karen Feigh, and Matthew Gombolay. Mixed-initiative human-robot teaming under suboptimality with online bayesian adaptation, 2024. URL https://arxiv.org/abs/2403.16178.

Anh Nguyen and Stefan Lee. Language-conditioned world model improves policy generalization by reading environmental descriptions. *arXiv preprint arXiv:2511.22904*, 2025.

Daniel Nyga, Subhro Roy, Rohan Paul, Daehyung Park, Mihai Pomarlan, Michael Beetz, and Nicholas Roy. Grounding robot plans from natural language instructions with incomplete world knowledge. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 714–723. PMLR, 29–31 Oct 2018. URL `https://proceedings.mlr.press/v87/nyga18a.html`.

OpenAI. Introducing GPT-Live, July 2026. URL `https://openai.com/index/introducing-gpt-live/`. Accessed: 2026-07-09.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand, 2019.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *NeurIPS*, 35:27730–27744, 2022.

Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

Aishwarya Padmakumar and Raymond J. Mooney. Dialog policy learning for joint clarification and active learning queries. In *The AAAI Conference on Artificial Intelligence (AAAI)*, February 2021. URL http://www.cs.utexas.edu/users/ai-labpub-view.php?PubID=127831.

Aishwarya Padmakumar, Peter Stone, and Raymond Mooney. Learning a policy for opportunistic active learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1347–1357, 2018a.

Aishwarya Padmakumar, Peter Stone, and Raymond J. Mooney. Learning a policy for opportunistic active learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP-18)*, Brussels, Belgium, November 2018b. URL http://www.cs.utexas.edu/users/ai-labpub-view.php?PubID=127713.

Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tur. Teach: Task-driven embodied agents that chat. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2017–2025, 2022.

Rohan Paleja, Michael Munje, Kimberlee Chang, Reed Jensen, and Matthew Gombolay. Designs for enabling collaboration in human-machine teaming via interactive and explainable systems, 2024. URL https://arxiv.org/abs/2406.05003.

Yingwei Pan, Zhaofan Qiu, Ting Yao, Houqiang Li, and Tao Mei. To create what you tell: Generating videos from captions. *Proceedings of the 25th ACM international conference on Multimedia*, 2017. URL https://api.semanticscholar.org/CorpusID:5039505.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

Jannik Peters, Constantin Waubert de Puiseau, Hasan Tercan, Arya Gopikrishnan, Gustavo Adolpho Lucas de Carvalho, Christian Bitter, and Tobias Meisen. Emergent language: a survey and taxonomy. *Autonomous Agents and Multi-Agent Systems*, 39(1), March 2025. ISSN 1573-7454. doi: 10.1007/s10458-025-09691-y. URL `http://dx.doi.org/10.1007/s10458-025-09691-y`.

Dean Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Conference on Neural Information Processing Systems (NeurIPS)*, 1988.

Bharat Prakash, Nicholas R Waytowich, Ashwinkumar Ganesan, Tim Oates, and Tinoosh Mohsenin. Guiding safe reinforcement learning policies using structured language constraints. In *SafeAI@ AAAI*, pages 153–161, 2020.

Kun Qian, Ahmad Beirami, Satwik Kottur, Shahin Shayandeh, Paul Crook, Alborz Geramifard, Zhou Yu, and Chinnadhurai Sankar. Database search results disambiguation for task-oriented dialog systems, 2021. URL `https://arxiv.org/abs/2112.08351`.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, pages 416–426. PMLR, 2023.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations, 2018. URL `https://arxiv.org/abs/1709.10087`.

Shreyas Sundara Raman, Vanya Cohen, David Paulius, Ifrah Idrees, Eric Rosen, Ray Mooney, and Stefanie Tellex. Cape: Corrective actions from precondition errors using large language models. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023.

Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11157–11166, 2020.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024. URL `https://arxiv.org/abs/2408.00714`.

Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, 2016. URL `https://api.semanticscholar.org/CorpusID:1563370`.

Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha Majumdar. Robots that ask for help: Uncertainty alignment for large language model planners, 2023. URL https://arxiv.org/abs/2307.01928.

Tianhe Ren, Qing Jiang, Shilong Liu, Zhaoyang Zeng, Wenlong Liu, Han Gao, Hongjie Huang, Zhengyu Ma, Xiaoke Jiang, Yihao Chen, Yuda Xiong, Hao Zhang, Feng Li, Peijun Tang, Kent Yu, and Lei Zhang. Grounding dino 1.5: Advance the "edge" of open-set object detection, 2024.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2016. URL https://arxiv.org/abs/1505.05770.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

Andres Rosero, Faustina Dinh, Ewart J. de Visser, Tyler Shaw, and Elizabeth Phillips. Two many cooks: Understanding dynamic human-agent team communication and perception using overcooked 2, 2021. URL https://arxiv.org/abs/2110.03071.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011. URL https://arxiv.org/abs/1011.0686.

Andrei Rusu, Sergio Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Krikpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295v2*, 2015.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. Crossing the reality gap: a survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, PP:1–1, 2021. URL `https://api.semanticscholar.org/CorpusID:243882665`.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilroberta, 2020. URL `https://huggingface.co/distilroberta-base`.

Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards – just map them to actions, 2020. URL `https://arxiv.org/abs/1912.02875`.

Eike Schneiders, Christopher Fourie, Stanley Celestin, Julie Shah, and Malte Jung. Understanding entrainment in human groups: Optimising human-robot collaboration from lessons learned during human-human collaboration. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, page 1–13. ACM, May 2024. doi: 10.1145/3613904.3642427. URL `http://dx.doi.org/10.1145/3613904.3642427`.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017a. URL `https://arxiv.org/abs/1502.05477`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017b. URL `https://arxiv.org/abs/1707.06347`.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*, 2020.

Mario Selvaggio, Marco Cognetti, Stefanos Nikolaidis, Serena Ivaldi, and Bruno Siciliano. Autonomy in physical human-robot interaction: A brief survey. *IEEE RA-L*, 6(4):7989–7996, 2021.

Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *Conference on Robot Learning*, pages 1332–1344. PMLR, 2023.

Rutav Shah, Roberto Martín-Martín, and Yuke Zhu. Mutex: Learning unified policies from multimodal task specifications. *arXiv preprint arXiv:2309.14320*, 2023.

Rutav Shah, Albert Yu, Yifeng Zhu, Yuke Zhu, and Roberto Martín-Martín. Bumble: Unifying reasoning and acting with vision-language models for building-wide mobile manipulation. *ICRA*, 2025.

Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. Videovla: Video generators can be generalizable robot manipulators. *Advances in neural information processing systems*, 38:95597–95621, 2026.

Afagh Mehri Shervedani, Siyu Li, Natawut Monaikul, Bahareh Abbasi, Barbara Di Eugenio, and Milos Zefran. Multimodal reinforcement learning for robots collaborating with humans, 2024. URL https://arxiv.org/abs/2303.07265.

Lucy Xiaoyang Shi, Zheyuan Hu, Tony Z. Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn. Yell at your robot: Improving on-the-fly from language corrections, 2024. URL https://arxiv.org/abs/2403.12910.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models, 2025. URL https://arxiv.org/abs/2502.19417.

Yi-Jen Shih, Desh Raj, Chunyang Wu, Wei Zhou, SK Bong, Yashesh Gaur, Jay Mahadeokar, Ozlem Kalinli, and Mike Seltzer. Can speech llms think while listening?, 2025. URL https://arxiv.org/abs/2510.07497.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. *Conference on Robot Learning*, 2022.

Andrew Silva, Nina Moorman, William Silva, Zulfiqar Zaidi, Nakul Gopalan, and Matthew Gombolay. Lancon-learn: Learning with language to enable generalization in multi-task manipulation. In *IEEE Robotics and Automation Letters*, 2021.

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

Sukriti Singh, Anusha Srikanthan, Vivek Mallampati, and Harish Ravichandar. Concurrent constrained optimization of unknown rewards for multi-robot task allocation, 2023. URL `https://arxiv.org/abs/2305.15288`.

Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. In *Robotics Science and Systems (RSS)*, 2020.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and A. Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218, 2014. URL `https://api.semanticscholar.org/CorpusID:2317858`.

Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. *arXiv preprint arXiv:2102.06177*, 2021.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *ArXiv*, abs/1908.08530, 2019. URL `https://api.semanticscholar.org/CorpusID:201317624`.

Alane Suhr, Claudia Yan, Charlotte Schluger, Stanley Yu, Hadi Khader, Marwa Mouallem, Iris Zhang, and Yoav Artzi. Executing instructions in situated collaborative interactions, 2022. URL https://arxiv.org/abs/1910.03655.

Chen Sun, Austin Myers, Carl Vondrick, Kevin P. Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7463–7472, 2019. URL https://api.semanticscholar.org/CorpusID:102483628.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, second edition, 2018.

Hao Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In *Conference on Empirical Methods in Natural Language Processing*, 2019. URL https://api.semanticscholar.org/CorpusID:201103729.

Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, 2009. URL https://api.semanticscholar.org/CorpusID:17216004.

DeepMind Interactive Agents Team, Josh Abramson, Arun Ahuja, Arthur Brussee, Federico Carnevale, Mary Cassin, Felix Fischer, Petko Georgiev, Alex Goldin, Mansi Gupta, Tim Harley, Felix Hill, Peter C Humphreys, Alden Hung, Jessica Landon, Timothy Lillicrap, Hamza Merzic, Alistair Muldal, Adam Santoro, Guy Scully, Tamara von Glehn, Greg Wayne, Nathaniel Wong, Chen Yan, and Rui Zhu. Creating multimodal interactive agents with imitation and self-supervised learning, 2022. URL https://arxiv.org/abs/2112.03763.

Open-X-Embodiment Team. Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration0. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, 2023. URL https://api.semanticscholar.org/CorpusID:263626099.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *arXiv preprint arXiv:1707.04175*, 2017.

Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. Learning to interpret natural language commands through human-robot dialog. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 1923–1929. AAAI Press, 2015. ISBN 9781577357384.

Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Justin Hart, Peter Stone, and Raymond J Mooney. Opportunistic active learning for grounding natural language descriptions. In *Conference on robot learning*, pages 67–76. PMLR, 2017a.

Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Justin Hart, Peter Stone, and Raymond J. Mooney. Opportunistic active learning for grounding natural language descriptions. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning (CoRL-17)*, pages 67–76, Mountain View, California, November 2017b. PMLR. URL `http://www.cs.utexas.edu/users/ai-labpub-view.php?PubID=127657`.

Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Nick Walker, Yuqian Jiang, Harel Yedidsion, Justin Hart, Peter Stone, and Raymond J. Mooney. Jointly improving parsing and perception for natural language commands through human-robot dialog. In *RSS Workshop on Models and Representations for Natural Human-Robot Communication (MRHRC-18). Robotics: Science and Systems (RSS)*, June 2018. URL `http://www.cs.utexas.edu/users/ai-labpub-view.php?PubID=127719`.

Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Nick Walker, Yuqian Jiang, Harel Yedidsion, Justin Hart, Peter Stone, and Raymond J. Mooney.

Improving grounded natural language understanding through human-robot dialog. In *IEEE International Conference on Robotics and Automation (ICRA)*, Montreal, Canada, May 2019a. URL http://www.cs.utexas.edu/users/ai-labpub-view.php?PubID=127746.

Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Nick Walker, Yuqian Jiang, Harel Yedidsion, Justin W. Hart, Peter Stone, and Raymond J. Mooney. Improving grounded natural language understanding through human-robot dialog. *ICRA*, pages 6934–6941, 2019b. URL https://api.semanticscholar.org/CorpusID:67703525.

Adam J. Thorpe, Stepan Tretiakov, Cheng-Hsi Hsiao, Su Ann Low, Xingjian Li, Hassan Iqbal, Neel P. Bhatt, Ufuk Topcu, and Krishna Kumar. Physically viable world models: A case for query-conditioned embodied ai, 2026. URL https://arxiv.org/abs/2605.30542.

Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. In *International Conference on Learning Representations*, volume 2025, pages 92033–92052, 2025.

Chenrui Tie, Shengxiang Sun, Yudi Lin, Yanbo Wang, Zhongrui Li, Zhouhan Zhong, Jinxuan Zhu, Yiman Pang, Haonan Chen, Junting Chen, et al. Manual2skill++: Connector-aware general robotic assembly from instruction manuals via vision-language models. *arXiv preprint arXiv:2510.16344*, 2025a.

Chenrui Tie, Shengxiang Sun, Jinxuan Zhu, Yiwei Liu, Jingxiang Guo, Yue Hu, Haonan Chen, Junting Chen, Ruihai Wu, and Lin Shao. Manual2skill: Learning to read manuals and acquire robotic skills for furniture assembly using vision-language models. *arXiv preprint arXiv:2502.10090*, 2025b.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

Marcel Torne, Anubha Mahajan, Abhijnya Bhat, and Chelsea Finn. Freeform preference learning for robotic manipulation, 2026. URL `https://arxiv.org/abs/2606.32027`.

M$^{\underline{a}}$ Mercedes Lopez Torné, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *ArXiv*, abs/2403.03949, 2024. URL `https://api.semanticscholar.org/CorpusID:268253222`.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance, 2014.

Shivam Vats, Oliver Kroemer, and Maxim Likhachev. Synergistic scheduling of learning and allocation of tasks in human-robot teams, 2022. URL `https://arxiv.org/abs/2203.07478`.

Manuela Veloso, Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal. Cobots: robust symbiotic autonomous mobile service robots. In *IJCAI, 2015*, page 4423–4429. AAAI Press, 2015. ISBN 9781577357384.

Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence – video to text. *2015*

IEEE International Conference on Computer Vision (ICCV), pages 4534–4542, 2015. URL https://api.semanticscholar.org/CorpusID:4228546.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and D. Erhan. Show and tell: A neural image caption generator. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, 2014. URL https://api.semanticscholar.org/CorpusID:1169492.

Homer Walke, Jonathan Yang, Albert Yu, Aviral Kumar, Jedrzej Orbik, Avi Singh, and Sergey Levine. Don't start from scratch: Leveraging prior data to automate robotic reinforcement learning. In *Proceedings of the 6th Conference on Robot Learning (CORL)*, 2022.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

Weikang Wan, Yifeng Zhu, Rutav Shah, and Yuke Zhu. Lotus: Continual imitation learning for robot manipulation through unsupervised skill discovery. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 537–544. IEEE, 2024.

Che Wang, Xufang Luo, Keith Ross, and Dongsheng Li. Vrl3: A data-driven framework for visual deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:32974–32988, 2022a.

Dian Wang, Mingxi Jia, Xupeng Zhu, Robin Walters, and Robert Platt. On-robot learning with equivariant models, 2022b. URL https://arxiv.org/abs/2203.04923.

Dian Wang, Robin Walters, and Robert Platt. SO(2)-equivariant reinforcement learning, 2022c. URL https://arxiv.org/abs/2203.04439.

Huaxiaoyue Wang, Kushal Kedia, Juntao Ren, Rahma Abdullah, Atiksh Bhardwaj, Angela Chao, Kelly Y Chen, Nathaniel Chin, Prithwish Dan, Xinyi Fan, Gonzalo Gonzalez-Pumariega, Aditya Kompella, Maximus Adrian Pace, Yash Sharma, Xiangwan Sun, Neha Sunkara, and Sanjiban Choudhury. Mosaic: A modular system for assistive and interactive cooking, 2024. URL https://arxiv.org/abs/2402.18796.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Yiping Lu, Zhengyuan Yang, Lijuan Wang, et al. Vagen: Reinforcing world model reasoning for multi-turn vlm agents. *Advances in Neural Information Processing Systems*, 38:172871–172933, 2026a.

Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models. *ArXiv*, abs/2310.01361, 2023. URL https://api.semanticscholar.org/CorpusID:263605851.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

Ziyan Wang, Meng Fang, Tristan Tomilin, Fei Fang, and Yali Du. Safe multi-agent reinforcement learning with natural language constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 38007–38015, 2026b.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *International Conference on Machine Learning*, 2007a. URL https://api.semanticscholar.org/CorpusID:6225453.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pages 1015–1022, 2007b.

Mark Woodward, Chelsea Finn, and Karol Hausman. Learning to interactively learn and assist, 2019. URL `https://arxiv.org/abs/1906.10187`.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, volume 2024, pages 10641–10662, 2024.

Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. Tidybot: Personalized robot assistance with large language models. *Autonomous Robots*, 2023a.

Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. Collabllm: From passive responders to active collaborators. In *ICML*, 2025.

Yue Wu, Yewen Fan, Paul Pu Liang, Amos Azaria, Yuanzhi Li, and Tom M Mitchell. Read and reap the rewards: Learning to play atari with the help of instruction manuals. *Advances in Neural Information Processing Systems*, 36: 1009–1023, 2023b.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11097–11107, 2020.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Reward shaping with language models for reinforcement learning. In *International Conference on Learning Representations*, 2023. URL https://api.semanticscholar.org/CorpusID:270063618.

Jingkai Xu and Xiangli Nie. Speci: Skill prompts based hierarchical continual imitation learning for robot manipulation. *IEEE Transactions on Cognitive and Developmental Systems*, 18:488–503, 2025. URL https://api.semanticscholar.org/CorpusID:277993717.

Zhiyuan Xu, Kun Wu, Zhengping Che, Jian Tang, and Jieping Ye. Knowledge transfer in multi-task deep reinforcement learning for continuous control. *arXiv preprint arXiv:2010.07494*, 2020.

Jonathan Yang, Catherine Glossop, Arjun Bhorkar, Dhruv Shah, Quan Vuong, Chelsea Finn, Dorsa Sadigh, and Sergey Levine. Pushing the limits of cross-embodiment learning for manipulation and navigation. *ArXiv*, abs/2402.19432, 2024. URL https://api.semanticscholar.org/CorpusID:268091245.

Jonathan Yang, Chelsea Finn, and Dorsa Sadigh. Data analogies enable efficient cross-embodiment transfer. *ArXiv*, abs/2603.06450, 2026. URL https://api.semanticscholar.org/CorpusID:286367896.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *arXiv preprint arXiv:2003.13661*, 2020.

Tsung-Yen Yang, Michael Y Hu, Yinlam Chow, Peter J Ramadge, and Karthik Narasimhan. Safe reinforcement learning with natural language constraints. *Advances in Neural Information Processing Systems*, 34:13794–13808, 2021.

L. Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Joseph Pal, H. Larochelle, and Aaron C. Courville. Describing videos by exploiting temporal structure. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4507–4515, 2015. URL https://api.semanticscholar.org/CorpusID:623318.

Yuanqi Yao, Siao Liu, Haoming Song, Delin Qu, Qizhi Chen, Yan Ding, Bin Zhao, Zhigang Wang, Xuelong Li, and Dong Wang. Think small, act big: Primitive prompt learning for lifelong robot manipulation. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22573–22583, 2025. URL https://api.semanticscholar.org/CorpusID:277468280.

Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy, 2020.

Albert Yu and Raymond J Mooney. Using both demonstrations and language instructions to efficiently learn robotic tasks. *arXiv preprint arXiv:2210.04476*, 2022.

Albert Yu, Adeline Foote, Raymond Mooney, and Roberto Martín-Martín. Natural language can help bridge the sim2real gap. In *Robotics: Science and Systems (RSS), 2024*, 2024.

Albert Yu, Chengshu Li, Luca Macesanu, Arnav Balaji, Ruchira Ray, Raymond Mooney, and Roberto Mart'in-Mart'in. Mixed-initiative dialog for human-robot collaborative manipulation. *ArXiv*, abs/2508.05535, 2025. URL https://api.semanticscholar.org/CorpusID:280546031.

Tian Yu, Jing Huang, and Qing Chang. Optimizing task scheduling in human-robot collaboration with deep multi-agent reinforcement learning. *Journal of Manufacturing Systems*, 60:487–499, 2021a. ISSN 0278-6125. doi: https://doi.org/10.1016/j.jmsy.2021.07.015. URL https://www.sciencedirect.com/science/article/pii/S0278612521001527.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019. URL `https://arxiv.org/abs/1910.10897`.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.

Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Sergey Levine, and Chelsea Finn. Conservative data sharing for multi-task offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021b.

Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, Brian Ichter, Ted Xiao, Peng Xu, Andy Zeng, Tingnan Zhang, Nicolas Heess, Dorsa Sadigh, Jie Tan, Yuval Tassa, and Fei Xia. Language to rewards for robotic skill synthesis, 2023a. URL `https://arxiv.org/abs/2306.08647`.

Xiao Yu, Maximillian Chen, and Zhou Yu. Prompt-based monte-carlo tree search for goal-oriented dialogue policy planning, 2023b. URL `https://arxiv.org/abs/2305.13660`.

Zhecheng Yuan, Zhengrong Xue, Bo Yuan, Xueqian Wang, Yi Wu, Yang Gao, and Huazhe Xu. Pre-trained image encoder for generalizable visual reinforcement learning. *Advances in Neural Information Processing Systems*, 35:13022–13037, 2022.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Proceedings of the 4th Conference on Robot Learning (CoRL)*, 2020.

Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv*, 2022.

Lihan Zha, Yuchen Cui, Li-Heng Lin, Minae Kwon, Montserrat Gonzalez Arenas, Andy Zeng, Fei Xia, and Dorsa Sadigh. Distilling and retrieving generalizable knowledge for robot manipulation via language corrections. In *2024 IEEE international conference on robotics and automation (ICRA)*, pages 15172–15179. IEEE, 2024.

Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. *CoRR*, abs/2111.07991, 2021. URL `https://arxiv.org/abs/2111.07991`.

Alex Zhang, Khanh Nguyen, Jens Tuyls, Albert Lin, and Karthik Narasimhan. Language-guided world models: A model-based approach to ai control. In *Proceedings of the 4th Workshop on Spatial Language Understanding and Grounded Communication for Robotics (SpLU-RoboNLP 2024)*, pages 1–16, 2024.

Jiahao Zhang, Anoop Cherian, Cristian Rodriguez, Weijian Deng, and Stephen Gould. Manual-pa: Learning 3d part assembly from instruction diagrams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6304–6314, 2025a.

Jian Zhang, Hanbo Zhang, Anxing Xiao, and David Hsu. Robot operation of home appliances by reading user manuals. *arXiv preprint arXiv:2505.20424*, 2025b.

Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. Up-vla: A unified understanding and prediction model for embodied agent. *arXiv preprint arXiv:2501.18867*, 2025c.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too?, 2018. URL `https://arxiv.org/abs/1801.07243`.

Michelle Zhao, Henny Admoni, Reid Simmons, Aaditya Ramdas, and Andrea Bajcsy. Conformalized interactive imitation learning: Handling expert shift and intermittent feedback. In *International Conference on Learning Representations*, volume 2025, pages 9238–9251, 2025.

Tony Zhao, Siddharth Karamcheti, Thomas Kollar, Chelsea Finn, and Percy Liang. What makes representation learning from videos hard for control? 2022. URL `https://api.semanticscholar.org/CorpusID:252635608`.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, 2020. URL `https://api.semanticscholar.org/CorpusID:221971078`.

Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. Rtfm: Generalising to novel environment dynamics via reading. *arXiv preprint arXiv:1910.08210*, 2019.

Bin Zhu, Bin Lin, Munan Ning, Yang Yan, Jiaxi Cui, HongFa Wang, Yatian Pang, Wenhao Jiang, Junwu Zhang, Zongwei Li, et al. Languagebind: Ex-

tending video-language pretraining to n-modality by language-based semantic alignment. *arXiv preprint arXiv:2310.01852*, 2023.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025a.

Xupeng Zhu, Yu Qi, Yizhe Zhu, Robin Walters, and Robert Platt. Equact: An se(3)-equivariant multi-task transformer for open-loop robotic manipulation, 2025b. URL `https://arxiv.org/abs/2505.21351`.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

Vita

Albert Yu was born and raised in Texas and California. He went to Tesoro High School ('17) in Orange County, California, where he enjoyed science, math, and history-related extracurriculars. Albert received a BS in Electrical Engineering and Computer Science from UC Berkeley ('21), where he worked as an undergraduate research assistant for Prof. Sergey Levine on real-robot offline RL. For four semesters, he served as a TA, and later head TA, for the Introductory AI course. Albert began his PhD at UT Austin in 2021, co-advised by Profs. Ray Mooney and Roberto Martín-Martín, during which he conducted the research of this thesis.

Address: albertyu101@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.