Title

Advancing the Cognitive Abilities of Embodied Agents: Large-Scale Simulations and Multi-Agent Collaborations

Author

Gong, Ran

Publication Date

2024

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Advancing the Cognitive Abilities of Embodied Agents: Large-Scale Simulations and Multi-Agent Collaborations

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Ran Gong

2024

ABSTRACT OF THE DISSERTATION

Advancing the Cognitive Abilities of Embodied Agents:
Large-Scale Simulations and Multi-Agent Collaborations

by

Ran Gong
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2024
Professor Demetri Terzopoulos, Co-Chair
Professor Song-Chun Zhu, Co-Chair

To construct a general artificial intelligence system, embodied agents must be able to perceive their environment, understand human language, engage in complex reasoning, manipulate objects, and collaborate with humans and each other. Cognitive science research suggests that intelligence emerges from sensorimotor experiences and interactions with the physical world. However, learning active perception and sensorimotor control through interaction with the physical environment can be challenging because existing algorithms are too slow for real-time learning, and embodied agents are fragile and expensive. Consequently, there is a pressing need for virtual simulation systems that can mimic complex behaviors and facilitate agent-environment interactions. In addition to mastering basic physical skills, embodied agents also need to engage in long-horizon task planning, coordination, and abstract reasoning to be effective in real-world scenarios.

The first line of research reported in this thesis focuses on developing simulation environments in which robots can interact with human users and their surroundings. We introduce a new simulation environment, `VRKitchen`, which enables the simulation of complex high-level behaviors and state changes. We also collect a dataset featuring human-environment interactions to predict human intentions. Furthermore, we develop a new system, `ARNOLD`, to simulate intricate low-level physics, including articulated objects

and liquids. Using the ARNOLD Dataset, we assess the abilities of robots to comprehend human language and execute complex manipulations under varied visual conditions, thereby evaluating their generalization capabilities in diverse and novel environments.

The second line of research addresses multi-agent collaboration and task allocation. We examine how robots of various types can cooperate with each other or with human users to accomplish common tasks. Initially, we propose a joint mind modeling framework based on the theory of mind to enhance the collaboration between humans and robots. Subsequently, we create a suite of multi-robot vision-based collaboration tasks, LEMMA, where robots positioned around a tabletop must collaborate to complete a task based on high-level instructions and also utilize tools. Lastly, leveraging large language models, we introduce a centralized multi-agent dispatcher framework, MindAgent, and its associated benchmarks and infrastructures.

The dissertation of Ran Gong is approved.

Bolei Zhou

Kai-Wei Chang

Ying Nian Wu

Song-Chun Zhu, Committee Co-Chair

Demetri Terzopoulos, Committee Co-Chair

University of California, Los Angeles

2024

To my family and friends who made this possible

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am deeply thankful to my family for their unwavering support throughout this journey. I owe immense gratitude to my mother, Danli Duan, who made numerous sacrifices to help me reach this point. I also want to express my heartfelt appreciation to my godmother, Janet Wang, for her sacrifices and support. She provided me with a place to call "home" in Los Angeles, a haven to which I can always return whenever things don't go as planned. Additionally, I am grateful to my uncle, Dennis Fox, for his long-time support. He introduced me to various American sports, of which I have become a big fan, and taught me how to be a skilled home chef. Today, I am proud to call myself an expert steak chef.

I would like to express my sincere gratitude to my advisors, Professor Song-Chun Zhu, for his invaluable guidance and mentorship. I am equally grateful to Professor Demetri Terzopoulos for his enduring support and mentorship over the years. During my internship at Amazon Alexa AI, I had the fortunate opportunity to work with Dr. Gaurav S. Sukhatme, and at Microsoft Research Redmond, I was privileged to collaborate with Dr. Jianfeng Gao.

I am especially grateful for the opportunity to have worked with an extremely talented group of individuals. I would like to thank Baoxiong Jia, Xu Xie, Qing Li, Xiaojian Ma, Pan Lu, Yining Hong, Jiangyong Huang, Yizhou Zhao, Zane Durante, Bidipta Sarkar, Qian Long, Zhi Li, Yixin Zhu, Shu Wang, Muzhi Han, and Mark Edmonds for their insightful discussions and valuable collaborations.

I am fortunate to have worked in multiple research labs. In Professor Song-Chun Zhu's lab, I had the opportunity to work on extremely large projects, which honed my engineering skills and prepared me well for future research endeavors. Additionally, I learned the fundamentals of conducting research and identifying long-term, worthwhile problems. Working with Professor Demetri Terzopoulos helped me learn to pay extreme attention to details. Working with Dr. Gaurav Sukhatme encouraged me to explore a wide range of topics in robotics, fostering a strong interest in embodied AI research. This

experience also taught me to set realistic goals, create milestones, and deliver results. Lastly, my time with Dr. Jianfeng Gao's team taught me how to organize meetings and promote efficient collaboration.

Additionally, I would like to thank Dr. Siyuan Huang for numerous helpful discussions on choosing research topics and for inspiring me to tackle more challenging problems. I would also like to give a special shout-out to Dr. Xiaofeng Gao for his detailed guidance and longstanding support.

Last but not least, I would like to thank each committee member for their insightful advice and valuable discussions.

I would not be where I am today without the support of these incredible individuals. I am forever indebted to them for their guidance and encouragement.

VITA

2018	B.S. Computer Science and Engineering, UCLA.
2020	M.S. Computer Science, UCLA.
2022–present	Ph.D. Candidate in Computer Science, UCLA.
2021	Applied Scientist Intern, Amazon
2022-2023	Applied Scientist Intern, Amazon
2023-2024	Research Intern, Microsoft Research Redmond

PUBLICATIONS

(* indicates equal contribution)

R. Gong*, Q. Huang*, X. Ma*, H. Vo, Z. Durante, Y. Noda, Z. Zheng, D. Terzopoulos, F. Li, J. Gao. "MindAgent: Emerging Gaming Interaction." *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL Findings).* 2024.

R. Gong*, J. Huang*, Y. Zhao, H. Geng, X. Gao, Q. Wu, W. Ai, Z. Zhou, D. Terzopoulos, S.-C. Zhu, B. Jia, S. Huang. "ARNOLD: A Benchmark for Language-Grounded Task Learning With Continuous States in Realistic 3D Scenes." *In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 20483-20495.* 2023. (A short version won the Spotlight Award in the *CoRL Language and Robot Learning Workshop*, 2022)

R. Gong, X. Gao, Q. Gao, S. Shakiah, G. Thattai, G. S. Sukhatme. "LEMMA: Learning Language-Conditioned Multi-Robot Manipulation." *IEEE Robotics and Automation Letters (RA-L).* 2023.

X. Gao, Q. Gao, R. Gong, K. Lin, G. Thattai, G. S. Sukhatme. "DialFRED: Dialogue-Enabled Agents for Embodied Instruction Following." *IEEE Robotics and Automation*

Letters (RA-L). 2022.

Y. Hong, Q. Li, R. Gong, D. Ciao, S. Huang, S.-C. Zhu. "SMART: A Situation Model for Algebra Story Problems via Attributed Grammar." *In Proceedings of the AAAI conference on artificial intelligence, vol. 35, no. 14, pp. 13009-13017.* 2021

P. Lu*, R. Gong*, S. Jiang*, L. Qiu, S. Huang, X. Liang, S.-C. Zhu. "Inter-GPS: Interpretable Geometry Problem Solving with Formal Language and Symbolic Reasoning." *In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) pp. 6774–6786.* 2021 (Oral presentation)

X. Gao*, R. Gong*, Y. Zhao, S. Wang, T. Shu, and S.-C. Zhu. "Joint Mind Modeling for Explanation Generation in Complex Human-Robot Collaborative Tasks." *In 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pp. 1119-1126.* 2020.

Z. Nan, T. Shu, R. Gong, S. Wang, P. Wei, S.-C Zhu, and N Zheng. "Learning to Infer Human Attention in Daily Activities." *Pattern Recognition*, 103: 107314, 2020.

X. Gao, R. Gong, T. Shu, X. Xie, S. Wang, and S.-C. Zhu. "VRKitchen: an Interactive 3D Environment for Learning Real Life Cooking Tasks." *ICML Reinforcement Learning for Real Life Workshop*, 2019.

CHAPTER 1

Introduction

Building artificial intelligence (AI) agents has been the dream of many research scientists for decades. In the early days, the pioneering Dartmouth workshop coined the term "artificial intelligence." Subsequently, the field experienced rapid development on various fronts. The optimism was high, and expectations were even higher. In the 1970s, Marvin Minsky told Life Magazine, "From three to eight years, we will have a machine with the general intelligence of an average human being." Even though such efforts did not meet the public's expectations at the time, the field of AI quickly grew in the following decades. People from a wide range of backgrounds began to study AI from different angles, such as Computer Vision, Natural Language Processing, and Robotics.

Each field has achieved significant progress, facilitating a multitude of applications that influence our daily lives. Nevertheless, to construct an agent with Artificial General Intelligence (AGI), it is imperative to demonstrate intelligence across all domains. Recent advancements in cognitive science also indicate that intelligence arises from the interaction between an agent and its environment, as well as through sensorimotor activity (Smith and Gasser, 2005). Consequently, there is a pressing need to develop a system capable of engaging with the world by perceiving its environment, comprehending human language, and collaborating with other agents.

Building and deploying such systems directly in the real world is often cumbersome, as real robots are expensive, fragile, and slow. Additionally, different research laboratories often have unique setups for their robots and environments, making it difficult for other laboratories to reproduce similar results. The recent advancements in computer vision and natural language processing have been largely driven by extensive datasets (Deng

et al., 2009; Yao et al., 2007). These fields benefit from unified benchmarks that evaluate results and thereby promote progress. However, robotics lacks a similar unified effort. It is impractical for laboratories on different continents to share identical hardware resources. Early efforts studied active perception in autonomous agents populating physics-based virtual worlds (Terzopoulos and Rabie, 1995; Rabie and Terzopoulos, 2000; Qureshi and Terzopoulos, 2008); however, the simulation software excluded multi-contact simulations and the OpenGL-shaded-polygon rendering quality was insufficiently photorealistic for the purposes of present-day real-world applications. With the rapid advancement in physics simulation and rendering technologies, it is now opportune to revisit the approach with the overarching goal of tackling the greater challenges faced by real world perception-decision-action systems. Fast and accurate physical simulations will pave the way for embodied agents to interact with the world on previously infeasible scales as it has become possible to simulate years of experience in mere seconds, thus unlocking tremendous potential for such agents.

As Smith and Gasser (2005) suggest, the human learning process is inherently social. We do not learn in isolation; rather, we learn from our parents, teachers, and peers. Therefore, developing machines capable of collaborating with other AI systems and with human users is crucial for the development of embodied systems exhibiting AGI. With recent advancements in simulation technologies for the gaming industry, we can now investigate these issues within simulations in open-world video games. Relevant applications require a deep understanding of task settings, the dynamics of collaboration, and often involve long-term planning.

1.1 Contributions of the Thesis

This thesis proposes, implements, and demonstrates multiple computational frameworks designed to enhance the multi-agent collaboration capabilities of current real-world systems, utilizing both physics simulators and video game development platforms. More specifically, the thesis makes the following five primary contributions:

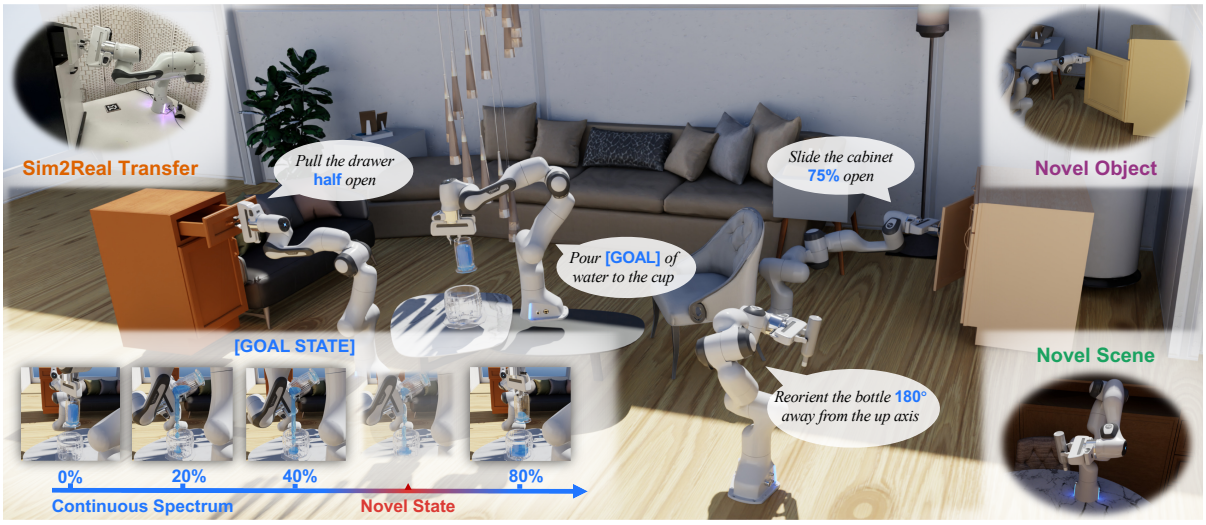


Figure 1.1: The ARNOLD benchmark for language-grounded task learning with continuous states in realistic 3D scenes. ARNOLD provides 8 tasks with their demonstrations for learning and a testbed for the generalization abilities of agents over (1) novel goal states, (2) novel objects, and (3) novel scenes.

1. One of the main challenges of applying reinforcement learning to real world applications is the lack of realistic and standardized environments for training and testing AI agents. We design and implement a Virtual Reality (VR) system, VRKitchen, with integrated functions that (i) enable embodied agents to perform real life cooking tasks involving a wide range of object manipulations and state changes and (ii) allow human teachers to provide demonstrations for training agents. We also provide standardized evaluation benchmarks and data collection tools to facilitate their broad use in research on learning real life tasks.[1]

2. Understanding the continuous states of objects is essential for task learning and planning in the real world. However, most existing task learning benchmarks assume discrete (*e.g.*, binary) object goal states, which poses challenges for the learning of complex tasks and transferring learned policy from simulated environments to the real world. Furthermore, state discretization limits a robot's ability to follow human instructions based on the grounding of actions and states. To tackle these challenges,

[1]Video demos, code, and data are available on the project website: https://sites.google.com/view/vr-kitchen/.

we present ARNOLD (Figure 1.1), a benchmark that evaluates language-grounded task learning with continuous states in realistic 3D scenes. ARNOLD is comprised of 8 language-conditioned tasks that involve understanding object states and learning policies for continuous goals. To promote language-instructed learning, we provide expert demonstrations with template-generated language descriptions. We assess task performance by utilizing the latest language-conditioned policy learning models. Our results indicate that current models for language-conditioned manipulations continue to experience significant challenges in novel goal-state generalizations, scene generalizations, and object generalizations. These findings highlight the need to develop new algorithms that address this gap and underscore the potential for further research in this area.[2]

3. Human collaborators can effectively communicate with their partners to finish a common task by inferring each other's mental states (*e.g.*, goals, beliefs, and desires). Such mind-aware communication minimizes the discrepancy among collaborators' mental states, and is crucial to the success of human ad-hoc teaming. We believe that robots collaborating with human users should demonstrate similar pedagogic behavior. Thus, we propose a novel eXplainable AI (XAI) framework for achieving human-like communication in human-robot collaborations, where the robot builds a hierarchical mind model of the human user and generates explanations of its own mind as a form of communication based on its online Bayesian inference of the user's mental state. To evaluate our framework, we conduct a user study on a real-time human-robot cooking task. Our experimental results show that the generated explanations of our approach significantly improve the collaboration performance and user perception of the robot.

4. Complex manipulation tasks often require robots with complementary capabilities to collaborate. We introduce a benchmark for LanguagE-conditioned Multi-robot MAnipulation (LEMMA) focused on task allocation and long-horizon object manipu-

[2]Project website: https://arnold-benchmark.github.io.

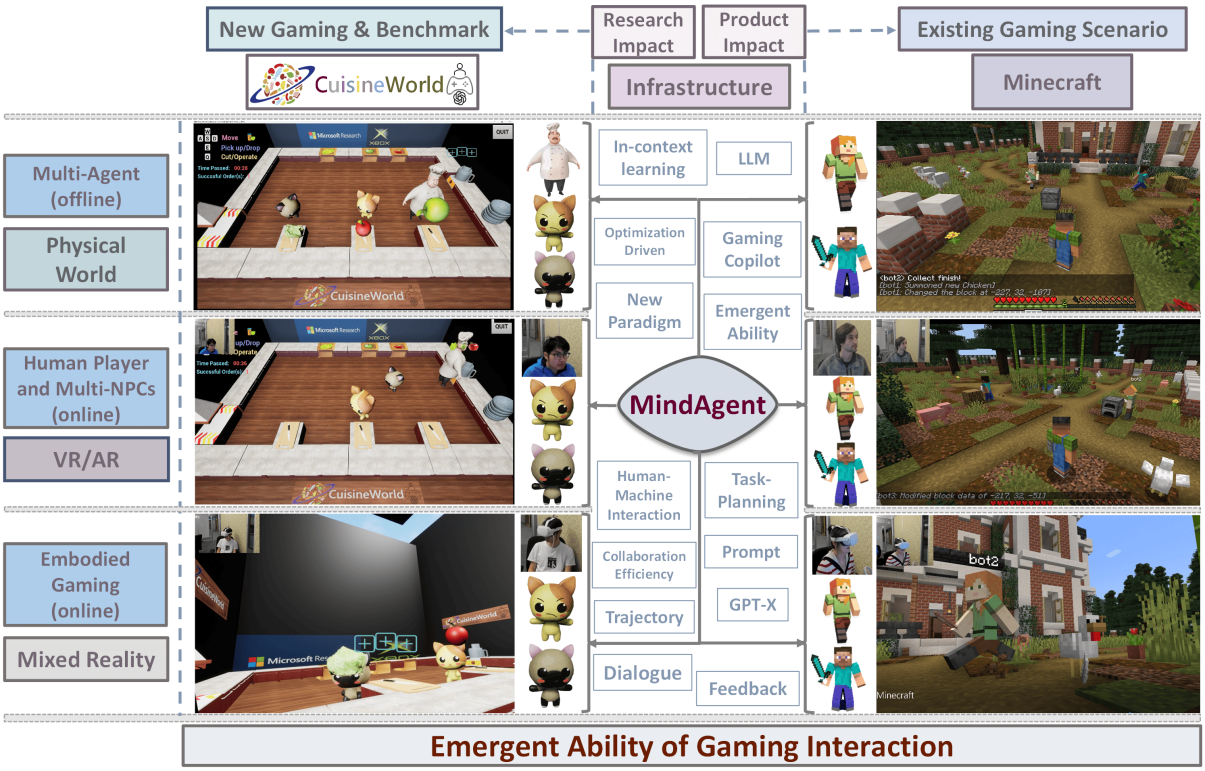


Figure 1.2: The `MindAgent` system for gaming interactions. `MindAgent` enables complex task planning in a multi-agent system and provides a human-AI collaboration infrastructure across various domains.

lation based on human language instructions in a tabletop setting. `LEMMA` features 8 types of procedurally generated tasks with varying degrees of complexity, some of which require the robots to use tools and pass tools to each other. For each task, we provide 800 expert demonstrations and human instructions for training and evaluations. `LEMMA` poses greater challenges compared to existing benchmarks, as it requires the system to identify each manipulator's limitations and assign sub-tasks accordingly while also handling strong temporal dependencies in each task. To address these challenges, we propose a modular hierarchical planning approach as a baseline. Our results highlight the potential of `LEMMA` for developing future language-conditioned multi-robot systems.

5. Large Foundation Models (LFMs) can perform complex scheduling in a multi-agent system and can coordinate agents to complete sophisticated tasks that require

extensive collaboration. However, despite the introduction of numerous gaming frameworks, the community lacks adequate benchmarks that support the implementation of a general multi-agent infrastructure encompassing collaboration between LFMs and human-NPCs. We propose a novel infrastructure, `MindAgent` (Figure 1.2), for evaluating planning and coordination capabilities in the context of gaming interaction. In particular, our infrastructure leverages an existing gaming framework to (i) act as the coordinator for a multi-agent system, (ii) collaborate with human players via instructions, and (iii) enable in-context learning based on few-shot prompting with feedback. Furthermore, we introduce `CuisineWorld`, a new gaming scenario and its related benchmark that supervises multiple agents playing the game simultaneously and measures multi-agent collaboration efficiency. We conduct comprehensive evaluations with a new auto-metric Collaboration Score (CoS) for assessing the collaboration efficiency. Finally, `MindAgent` is deployed in real-world gaming scenarios in a customized VR version of `CuisineWorld` and adapted in the "Minecraft" domain. Our work involving LFMs within our new infrastructure for general-purpose scheduling and coordination elucidates how such skills may be obtained by learning from large language corpora.

1.2 Outline of the Thesis

This thesis proposes simulation environments, benchmarks, user study protocols, and computational frameworks to address the challenges of embodied simulation, learning, and collaboration. Chapter 2 reviews prior work that is relevant to the range of topics spanned by the thesis research.

The technical body of the dissertation is in two parts. Part I, which focuses on simulation environments, comprises Chapter 3 and Chapter 4. Part II, which focuses on multi-agent collaboration, comprises Chapter 5, Chapter 6, and Chapter 7.

In Chapter 3, we develop a VR-based virtual kitchen (Gao et al., 2019; Nan et al., 2020). We provide a set of tools for collecting data using VR devices and web interfaces.

We collect synthetic indoor cooking datasets using our toolkits, and we benchmark the performances of various reinforcement learning agents. We further collect a multi-view dataset for human intention prediction.

In Chapter 4, we develop a photo-realistic and a physical-realistic simulation environment for robots (Gong et al., 2023b). We further provide a benchmark for language-grounded task learning on continuous object states. We perform a preliminary sim2real transfer test and show that our benchmark is valuable for sim2real transfer.

In Chapter 5, we develop a real-time collaborative cooking game to study human-robot collaborations (Gao et al., 2020). We propose a collaboration framework where human users and robots collaborate together to finish a joint task. We conduct a user study to confirm, in both subjective and objective measures, its effectiveness in collaboration.

In Chapter 6, we develop a multi-robot collaboration dataset and its associated environment based on lone-horizon task planning and manipulations (Gong et al., 2023a). We further improve the performance of the baseline models through a hierarchical modular model.

In Chapter 7, we develop a text-based gaming environment to study multi-agent collaborations and task planning (Gong et al., 2023c). We design an LLM based framework to improve collaboration efficiency. We conduct a user study to confirm the potential of collaborating with human players. We then transfer the framework to Minecraft and demonstrate its usability.

Finally, in Chapter 8, we draw the key conclusions from our research and discuss promising avenues for future work.

Appendix A and Appendix B present additional details about `ARNOLD` and `MindAgent`, respectively.

CHAPTER 2

Related Work

This chapter reviews prior work that is relevant to the range of topics spanned by the thesis research.

2.1 Simulation Platforms

Traditionally, visual representations are learned from static datasets. Either containing prerecorded videos (Rohrbach et al., 2012) or images (Deng et al., 2009), most of them fail to capture the dynamics in viewpoint and object state during human activities, in spite of their large scale. Some early systems (Qureshi and Terzopoulos, 2008; Rabie and Terzopoulos, 2000; Terzopoulos and Rabie, 1995; Lin et al., 2016) try to simulate the dynamics of human activities in order to support the development of smart visual surveillance systems and research into active computer vision for navigation. However, agents in the environment cannot be trained in a fined grained level for compositional tasks, and the environments often do not exhibit dynamic changes caused by the agents' actions.

To address this issue, there has been a growing trend to develop 3D virtual platforms for training embodied agents in dynamic environments. Typical systems include 3D game environments (Kempka et al., 2017; Beattie et al., 2016; Johnson et al., 2016), and robot control platforms (Todorov et al., 2012; Coumans and Bai, 2016; Fan et al., 2018; Plappert et al., 2018). While these systems offer physics simulation and 3D rendering, they fail to provide realistic environments and daily tasks humans face in the real world.

More recently, based on 3D scene datasets such as Matterport3D (Chang et al.,

Env.	Large-scale	Physics	Realistic	State	Manipulation	Avatar	Demon
Malmo (Johnson et al., 2016)	√			√			
DeepMind Lab (Beattie et al., 2016)							
VizDoom (Kempka et al., 2017)							
MINOS (Savva et al., 2017)	√		√				
HoME (Brodeur et al., 2017)	√	√	√				
Gibson (Xia et al., 2018)	√	√	√			√	
House3D (Wu et al., 2018)	√	√	√				
AI2-THOR (Kolve et al., 2017)		√	√	√			
VirtualHome (Puig et al., 2018)			√	√	√	√	
SURREAL (Fan et al., 2018)		√			√		√
`VRKitchen` (ours)		√	√	√	√	√	√

Table 2.1: Comparison with other 3D virtual environments. *Large-scale:* a large number of scenes. *Physics:* physics simulation. *Realistic:* photo-realistic rendering. *State:* changeable object states. *Manipulation:* enabling object interactions and manipulations. *Avatar:* humanoid virtual agents. *Demonstration:* user interface to collect human demonstrations.

2018) and SUNCG (Song et al., 2017), there are have been several systems simulating more realistic indoor environments (Brodeur et al., 2017; Wu et al., 2018; Savva et al., 2017; McCormac et al., 2017; Xia et al., 2018) for visual navigation tasks and basic object interactions such as pushing and moving funitures (Kolve et al., 2017). While the environments in these systems are indeed more realistic and scalable compared to previous systems, they still can not simulate complex object manipulation that are common in our daily life. Puig et al. (2018) took a step forward and has created a dataset of common household activities with a larger set of agent actions including pick-up, switch on/off, sit and stand-up. However, this system was only designed for generating data for video understanding. In contrast, our system emphasizes training and evaluating agents on virtual cooking tasks, which involves fine-grained object manipulation on the level of object parts (*e.g.*, grasping the handle of a knife), and flexible interfaces for allowing both human users and AI algorithms to perform tasks. Our system also simulates the animation of object state changes (such as the process of cutting a fruit) and the gestures of humanoid avatars (such as reaching for an object) instead of only showing pre-conditions and post-effects as in (Kolve et al., 2017). A detailed comparison between our system and other virtual environments is summarized in Table 2.1.

2.2 Imitation Learning

Learning from demonstration or imitation learning is proven to be an effective approach to train machine agents efficiently (Abbeel and Ng, 2004; Syed and Schapire, 2008; Ross et al., 2010). Collecting diverse expert demonstrations with 3D ground-truth information in real world is extremely difficult. We believe the VR interface in our system can greatly simplify and scale up the demonstration collection.

2.3 VR for AI

VR provides a convenient way to evaluate AI algorithms in tasks where interaction or human involvement is necessary. Researches have been conducted on many relevant domains, including physical intuition learning (Lerer et al., 2016), human-robot interaction (Liu et al., 2017a; de Giorgio et al., 2017), learning motor control from human demonstrations (Haidu et al., 2015; Kawasaki et al., 2001; Belousov et al., 2001). Researchers have also used VR to collect data and train computer vision models. To this end, several plugins for game engines have been released, such as UETorch (Lerer et al., 2016) and UnrealCV (Qiu and Yuille, 2016). To date, such plugins only offer APIs to control game state and record data, requiring additional packages to train virtual agents, or to gather data for other computer vision tasks.

2.4 Datasets for Computer Vision Tasks

Deng et al. (2009) started the big data era for modern computer vision research. With the increasing popularity of deep learning, there are a lot of datasets catering to different tasks. For human attention predictions, CAD120 (Koppula et al., 2013), has been commonly used. However, creating these dataset will require a significant amount of manpower to design tasks, record videos, and annotate data. Amazon Mechanical Turks are commonly used for annotation purposes; however, according to our own experiences, turker annotations are often noisy, so it takes a large amount of time for researchers to

design a protocol to make sure annotations gathered from turkers are reliable. For tasks like object recognition, researchers often want to label the smallest bounding box around the object; however, in real annotation scenarios, it is really hard for humans to find the smallest bounding box around the object. This might due to the fact that turkers are paid by the number of images they annotated not by how accurate their boxes are. As long as, their annotations are reasonable, researchers will often give them a pass. However, recent studies (Ratner et al., 2017) has demonstrated that inaccuracy in the data annotations can often produce meaningful differences in the final model. Therefore, it is crucial to have accurate annotations for the dataset. In a virtual environment, since we have all the information about the object model, it is relatively easy to obtain accurate ground truth information through transformation matrices and projection matrix.

2.5 Simulators for Embodied AI

Significant progress has recently been made in developing simulators for training and evaluating AI agents to perform indoor household activities (Li et al., 2021; Fu et al., 2022; Gan et al., 2021; Chen et al., 2022a; Deitke et al., 2022b; Chang et al., 2018). To mitigate complexity, most of these simulators make simplifications about world states and actions, abstracting robot manipulation into symbolic planning in discrete action (Puig et al., 2018; Kolve et al., 2017) and state spaces. However, agents trained in such settings are unaware of the relationship between actions and the geometries and dynamics of objects, therefore limiting their abilities in real-world scenarios. Recent efforts have gradually transitioned to continuous action spaces, but they still make some simplifications. For example, grasping is often simplified by attaching a nearby object to the gripper (Ehsani et al., 2021; Srivastava et al., 2022), or through contact (Li et al., 2022; James et al., 2020; Szot et al., 2021).

Among works that provide continuous object state change simulation, most do not focus on manipulating object states in a precise and fine-grained manner. For example, `VRKitchen` (Gao et al., 2019) defines task goals in a discrete manner even though the

underlying object states are continuous. Softgym (Lin et al., 2020) is an object manipulation benchmark that provides a realistic simulation of deformable objects; however, it lacks diversity among objects and scenes.

By contrast, ARNOLD provides a wide variety of scenes and objects. And ARNOLD simulates continuous states for articulated objects and simulates fluids at the particle level. We control the robots with 7-DOF continuous control and friction-based grasping powered by a state-of-the-art physics engine (PhysX 5.0). Whereas most of the environments (Gu et al., 2023; Szot et al., 2021) optimize for speed, we optimize for the realism of the rendering. And ARNOLD also equips a remarkable rendering speed at 185 FPS (37 FPS with five cameras).

2.6 Language Conditioned Manipulation

Relating human language to robot actions has been of recent interest (Lynch and Sermanet, 2021; Stepputtis et al., 2020; Zheng et al., 2022; Jiang et al., 2022; Deng et al., 2020; Huang et al., 2022b, 2023; Nair et al., 2022; Zeng et al., 2022). However, the environments in these efforts either lack realistic physics (Shridhar et al., 2020a, 2022a) or do not have realistic scenes (Shridhar et al., 2022b; Mees et al., 2022b) where the surroundings of the agent will constrain its motion, and different scene objects might occlude the agent's viewpoint. Additionally, systems like (Huang et al., 2022b; Brohan et al., 2022; Driess et al., 2023; Lynch et al., 2022) are application-based, lacking a systematic benchmark for language-conditioned manipulation. Most importantly, prior work aims to ground human language to static object properties, such as colors and shapes. By contrast, ARNOLD provides instructions for continuous object states. We compare between ARNOLD and other related benchmarks in Table 2.2. In addition, most studies typically use a single-robot setting. In (Tan et al., 2020; Liu et al., 2022c), a single agent can still finish the task although multiple agents are available. We investigate, with limited reachability, how multiple robots can collaborate with each other in LEMMA. Our task settings are designed to require the collaboration of two robots to successfully complete the task. In our setting,

Benchmark	Language	Multi Camera	Fluid	Physics	Continuous	Scenes	Robot	Rendering	Flexible Material	Generalization
Alfred (Shridhar et al., 2020a)	✓	✗	✗	✗	✗	✓	✗	R	✗	✗
Maniskill (Mu et al., 2021; Gu et al., 2023)	✗	✓	✓	✓	✓	✗	✓	R	✗	✓
Calvin (Mees et al., 2022b)	✓	✓	✗	✓	✗	✓	✓	R	✗	✓
Behavior (Li et al., 2022; Srivastava et al., 2022)	✗	✗	✓	✓	✓	✓	✓	RT	✓	✗
KitchenShift (Xing et al., 2021)	✗	✗	✗	✓	✗	✗	✓	R	✗	✓
RLBench (James et al., 2020)	✓	✓	✗	✓[1]	✗	✗	✓	R	✗	✗
Softgym (Lin et al., 2020)	✗	✗	✓	✓	✓	✗	✗	R	✗	✗
Orbit (Mittal et al., 2023)	✗	✗	✓	✓	✗	✓	✓	RT	✓	✗
Vlmbench (Zheng et al., 2022)	✓	✓	✗	✓[1]	✓	✗	✓	R	✗	✗
Ravens (Zeng et al., 2021; Shridhar et al., 2022a)	✓	✓	✗	✓	✗	✗	✓	R	✗	✓
Habitat HAB (Szot et al., 2021)	✗	✗	✗	✗	✗	✓	✓	R	✗	✓
TDW Transport (Gan et al., 2021, 2022)	✗	✗	✗	✗	✗	✓	✓	R	✗	✗
ARNOLD	✓	✓	✓	✓	✓	✓	✓	RT	✓	✓

Table 2.2: Comparison with existing benchmarks. ARNOLD features language-grounded robot control over continuous object states with a large number of demonstrations in photo-realistic scenes. ARNOLD also leverages advanced physics simulations powered by PhysX 5.0 to simulate articulated bodies and fluids. *Language:* Task goals are specified by human language instruction. *Multi-Camera:* Robot is equipped with multiple cameras. *Fluid:* Advanced fluid simulation. *Physics:* Realistic physics simulation with realistic grasping. [1]: RLbench-based benchmarks use simplified grasping. *Continuous:* Object state and goal state are continuous. *Scene:* Tasks are performed with a realistic scene background. *Robot:* Perform actions with real robots for all tasks. *R:* Rasterization. *RT:* RayTracing. *Flexible Material:* Easy to change materials and textures. *Generalization:* Systematic generalization test at different levels.

due to the limited workspace reachability of each robot, each target object can only be manipulated by one robot initially, making collaboration necessary to achieve the task goals.

2.7 Continuous State Understanding

Some recent research tries to predict object states (Liu et al., 2017b; Nagarajan and Grauman, 2018). However, the object states are discrete rather than continuous. More recently, researchers (Weng et al., 2021; Di et al., 2022; Wei et al., 2022a; Tseng et al., 2022) tried to predict object states continuously. However, they do not address the manipulation of objects from arbitrary starting states to the desired states. Moreover, they do not model the language grounding process. Most recently, Ma et al. (2023) propose a method to perform precise object state manipulations, but their approach does not perform language grounding and only a small subset of our tasks are covered with their simple motion primitives. Compared with prior work, we provide more diverse goal states to cover the continuous state space rather than learning only binary goal states.

This allows models to understand the continuous state space. Moreover, we also propose evaluation of generalization in terms of continuous state understanding (see Section 4.2.4). This evaluates how the model leverages its understanding of the state space to generalize within a continuous spectrum, which is rarely studied in prior works. Though more goal states can be added with our continuous simulation, we leave `ARNOLD` at the current scale since more states will lead to greater data generation costs due to the compositions of object/scene/state.

2.8 Human-Aware Planning

Designing robots that can work with humans has been widely studied by researchers. Most of the prior works hope to create robots to better understand and adapt to human collaborators. Liu et al. (2016) evaluate a collaborative task allocation framework based on a Bayesian inference of human intention. Hadfield-Menell et al. (2016) propose a formulation of the value alignment problem assuming the robot learning an unknown human reward function. Optimal solutions can be achieved when the human demonstrates active teaching behavior. To deal with sensor uncertainty and task ambiguity in a collaborative assembly task, Hawkins et al. (2014) use an And-Or tree structure as the task representation, which is similar to our approach. When sub-optimal user behavior are encountered, Reddy et al. (2018) propose to learn the incorrect human internal dynamics model via inverse RL and then perform an internal-to-real dynamics transfer to assist users in shared-autonomy tasks. Our framework differs from this line of research in that we also aim at improving humans' understanding of robots' models using communicative actions. Such two-way understanding will further help human-robot collaborations.

2.9 Goal-Driven Explainable AI

In contrast to data-driven XAI which improves understanding of "black-box" machine learning algorithms given input data, goal-directed XAI typically explains the behavior of

an agent or robot for a specific task (Langley et al., 2017; Anjomshoae et al., 2019; Miller, 2019), in order to increase model transparency (Struckmeier et al., 2019), human's trust (Wang et al., 2016) or task performance (Xu and Dudek, 2015). Some of the works achieve this aim by enabling robots to directly generate easy-to-understand motions (Dragan and Srinivasa, 2013; Kwon et al., 2018) or task plans (Zhang et al., 2017). Other works, similar to ours, focus on using explicit communication to change user mental state, *e.g.*, updating users' incorrect reward functions (Tabrez et al., 2019), correcting users' false belief or misunderstanding about the environment (Gong and Zhang, 2018; Sreedharan et al., 2018), resolving the disagreement between collaborators' actions (Nikolaidis et al., 2018) or providing users with necessary knowledge about the current situation (Devin and Alami, 2016). Compared to these work that often require offline training with humans or theoretical assumptions on the human models, this chapter takes a direct approach to generate explanations solely based on an online estimation of human model and knowledge of the task structure. The experiment results show our approach is empirically effective in an ad-hoc human-robot teaming settings (Stone et al., 2010) where pre-coordination is not available.

2.10 Visual Multi-Agent Collaboration

Visual multi-agent collaboration has attracted attention in recent embodied AI research (Tan et al., 2020; Jain et al., 2020, 2019; Wang et al., 2021; Chen et al., 2020; Liu et al., 2022b,c). However, among the works that involve object manipulation, simplified non-physics based atomic actions are often employed as an abstraction for manipulation. These works often use a magic glove to attach an object to the gripper as long as hand-crafted conditions are met. For example, an object within 15cm to the gripper can be automatically snapped to the gripper (Szot et al., 2021). While this simplification does not affect learning robot task planning, it is unsuitable for learning low-level manipulation policies. We do not make such a simplification here instead using a gripper to interact with objects physically. Additionally, most previous works study the collaboration problem in

a single-task setting. In contrast, our work LEMMA uses a multi-task setting requiring the comprehension of a textual description to understand the goal.

2.11 Bimanual Robot Manipulation

There is a rich set of literature on bimanual robot manipulation (Chen et al., 2022b; Takata et al., 2022; Stavridis et al., 2021; Smith et al., 2012; Zhang et al., 2019; Stepputtis et al., 2022; Lertkultanon and Pham, 2018). These works address important problems in dual-arm coordination with a focus on coordinated control and collision avoidance. However, there is less exploration of multi-robot task planning and allocation for long-horizon tasks with strong temporal dependencies, along with workspace management. In addition, these works typically do not involve vision and language inputs, especially for the recognition of the physical limitations of different robots from vision input. More importantly, previous research usually employs robot arms of the same type. In contrast, our work, LEMMA, considers the settings of both heterogeneous and homogeneous robot arms.

2.12 Visual Robot Task and Motion Planning

Traditionally, most works in this area use search over pre-defined domains for planning, which require extensive domain knowledge and accurate perception. Moreover, they often scale poorly with an increasing number of objects. Another line of work involves generating task and motion plans given scene images (Driess et al., 2020; Driess et al., 2021). In contrast, in our settings, the model uses both RGBD images and textual descriptions as input for multi-task learning. Most recently, Singh et al. (2023) generated robot policies in the form of code tokens using large language models (LLMs). Nonetheless, they only focus on single-agent task planning. We compare and contrast our benchmark with existing works in Table 2.3.

Benchmark	Language	Multi-task	Manipulation	Multi-agent	Tool Use	Temporal Dep.
Alfred (Shridhar et al., 2020a)	✓	✓	✗	✗	✓	✓
MQA (Deng et al., 2020)	✓	✓	✓	✗	✗	✗
Calvin (Mees et al., 2022b)	✓	✓	✓	✗	✗	✗
EQA (Tan et al., 2020)	✓	✓	✗	✓	✗	✗
Ravens (Zeng et al., 2021)	✗	✗	✓	✗	✗	✗
Vlmbench (Zheng et al., 2022)	✓	✗	✓	✗	✗	✗
CH-MARL (Sharma et al., 2022)	✓	✗	✗	✓	✗	✗
TBP (Jain et al., 2019)	✗	✗	✗	✓	✗	✗
EMATP (Liu et al., 2022c)	✓	✓	✗	✓	✓	✓
LEMMA	✓	✓	✓	✓	✓	✓

Table 2.3: Comparison with other benchmarks. LEMMA evaluates the performance of language-conditioned multi-agent object manipulation in long-horizon tasks. *Multi-task*: using a multi-task setting. *Language*: language instructions to specify goal. *Manipulation*: physical object manipulation. *Multi-agent*: requiring multiple agents for task completion. *Tool use*: requiring the robot to use a tool to interact with other objects. *Temporal Dep*: temporal dependency between sub-tasks.

2.13 Multi-Agent Coordination

The field of multi-agent collaboration boasts a comprehensive body of literature. Traditionally, such collaborations have been modeled using the MDP/POMDP frameworks (Lowe et al., 2017; Rashid et al., 2020; Jain et al., 2019; Wu et al., 2021; Gao et al., 2023). However, there has been a recent shift towards using LFMs for these collaborations. For instance, Zhang et al. (2023b) delved into how LFMs might communicate and cooperate in a watch-and-help (WAH) task. Meanwhile, Zhang et al. (2023a) investigated a two-agent collaboration game inspired by the simpler dynamics of the two-agent Overcooked-style game. Notably, their research mainly concentrated on the task success rate, with most studies typically anchored to a single task objective. By contrast, we emphasize the importance of collaboration efficiency in scenarios encompassing multiple task objectives. Further, our research uniquely focuses on evaluating the collaborative efficiency of two or more agents. Additionally, while other works such as that of Park et al. (2023); Wu et al. (2021) simulate each agent individually, we employ a centralized system. This not only significantly reduces the number of API calls but also reduces context length, making it more appropriate for use in gaming applications.

2.14 Planning With LFMs

A number of works leverage LFMs to perform task planning (Huang et al., 2022a; Wang et al., 2023a; Yao et al., 2023; Li et al., 2023; Wang et al., 2024), specifically the LFMs' WWW-scale domain knowledge and emergent zero-shot planning abilities to perform complex task planning and reasoning. Recent robotics research also leverages LFMs to perform task planning (Ahn et al., 2022; Huang et al., 2022b; Liang et al., 2022) by decomposing natural language instruction into a sequence of subtasks, either in the natural language form or in Python code , then using a low-level controller to execute these subtasks. Additionally, Huang et al. (2022b), Liang et al. (2022), and Wang et al. (2023b) also incorporate environmental feedback to improve task performance.

2.15 Benchmarks Using Games

Numerous games have been developed to study task planning (Baker et al., 2022; Carroll et al., 2019; Bakhtin et al., 2022), yet only a handful delve into multi-agent collaborations. Even within this limited subset, the focus predominantly remains on two-agent interactions where responsibilities are unevenly distributed between the agents (Wan et al., 2022; Puig et al., 2020)—it is common for one player to assume a dominant role while the other provides support. By contrast, our work assumes the equal apportion of responsibilities across agents, and we expand our investigation to encompass collaborations involving more than two agents, even including human players. While some previous studies have ventured into multi-task settings, none has delved into scenarios where agents must compete for resources to complete multiple distinct tasks with varied levels of difficulty within a single episode. Additionally, our work differs from that of Carroll et al. (2019) in that our game settings feature a diverse array of tools and task objectives, thereby generating an exponentially larger task space.

Part I

Simulation Environments

CHAPTER 3

VRKitchen: An Interactive 3D Virtual Environment for Task-Oriented Learning

3.1 Introduction

Thanks to the recent success in many domains of AI research, humans now have built machines that can accurately detect and recognize objects (Krizhevsky and Hinton, 2012; He et al., 2017), generate vivid natural images (Brock et al., 2018), and beat human Go champions (Silver et al., 2017). However, a truly intelligent machine agent should be able to solve a large set of complex tasks in the physical world by adapting itself to unseen surroundings and planning a long sequence of actions to reach the desired goals, which is still beyond the capacity of current machine models. To achieve state-of-the-art results, these systems often need a simulation environment which agents can interact with. However, in some domains, collecting datasets for agents is very expensive, slow, and often inaccurate. This gives rise to the need for an environment capable of synthesizing interactive datasets for different tasks. In particular, we are interested in the following two dataset-generation approaches for the present work.

3.1.1 Generating Multi-Modal Datasets in a Dynamic Environment

According to psychology studies (Smith and Gasser, 2005), humans learn from multi-modal inputs (Vision, Sound, Touches, *etc.*.). Different input modules self-teach each other so that infants can obtain a rich and compact experience about the world. Recent works (Ngiam et al., 2011) have been using sensor fusion-like algorithms to merge different

modalities of inputs. Therefore, an environment that can obtain different modalities of sensor inputs with interactivity should be designed and implemented.

Researchers (Smith and Gasser, 2005) also indicate that the learning experience for infants is physical, and infants often explore the environment to find out what task to learn and the solutions to these tasks. To better simulate the real-world scenario where the appearance of the same object may change dramatically as a result of actions (Isola et al., 2015; Fathi and Rehg, 2013; Liu et al., 2017b), the environment needs to have rich fluent changes. To capture such variation in object appearance, the agent is required to have a better visual representation of the environment dynamics. For example, the agent should recognize the tomato even if it is cut into pieces and put into a container. To acquire such visual knowledge, it is important for an agent to learn from physical interactions and reason over the underlying causality of object state changes. Therefore, it is critical to have an interaction-based dynamic world.

Long sequences of events are often needed for certain tasks: such as human attention prediction and human intention prediction. In order to perform these tasks, current systems often need a large amount of annotated data. Therefore, a system that is capable of generating a large number of highly customizable annotated data will potentially be helpful to the future research.

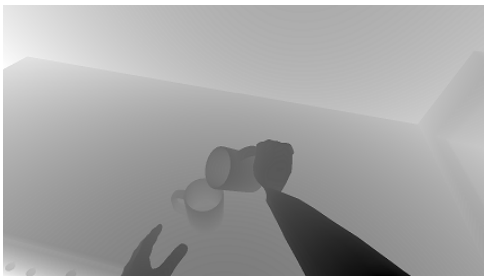
There have been work on implementing interaction-based learning in lab environments (Lerer et al., 2016; Agrawal et al., 2015; Haidu et al., 2015), but the limited scenarios greatly restrict scalability and reproducitibility of prior work, plus the ad-hoc environments often do not come with dataset generation capability. We believe that building a realistic simulation platform is a good alternative since i) the performance of different algorithms can be easily evaluated and benchmarked, and ii) a large set of diverse and realistic environments and tasks can be designed and customized. iii) customizable multi-modal data can be relatively easily generated (Figure 3.1, Figure 3.2, Figure 3.3).

(a) First subfigure

(b) Second subfigure

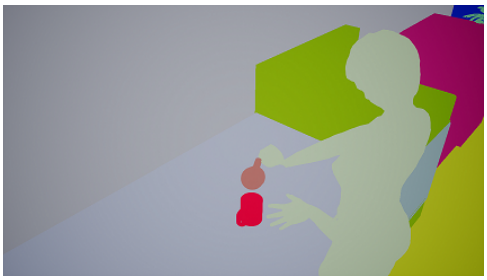
Figure 3.1: RGB

(a) First subfigure for Depth

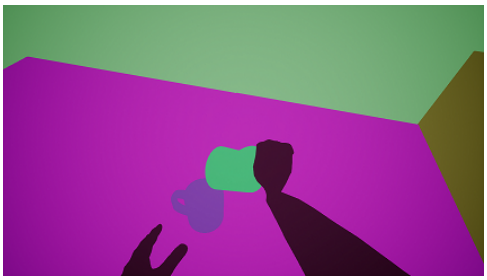


(b) Second subfigure for Depth

Figure 3.2: Depth



(a) First subfigure for Segmentation



(b) Second subfigure for Segmentation

Figure 3.3: Segmentation

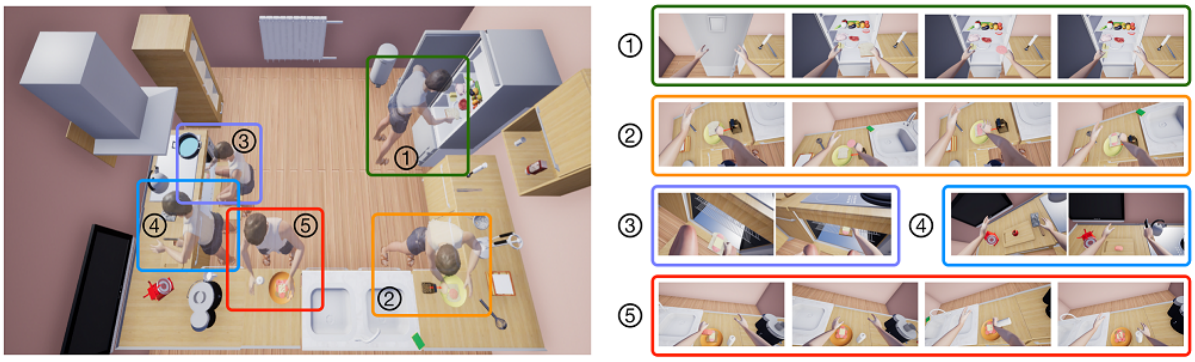


Figure 3.4: A sample sequence of an agent making a *sandwich*. Rectangles on the left graph represent five necessary sub-tasks, including (1) taking ingredients from the fridge, (2) putting ham and cheese on the bread, (3) using the oven, (4) cutting a tomato, and (5) adding some sauce. Each rectangle on the right graph indicates the atomic actions required to finish a sub-task.

3.1.2 Collecting Human Demonstrations to Bootstrap Agent Models

Training an agent from scratch is extremely difficult in complex environments. To bootstrap the training, it is common to let an agent imitate human experts by watching human demonstrations (Ng and Russell, 2000; Ziebart et al., 2008; Giusti et al., 2016). Previous work has shown that learning from demonstrations (or imitation learning) significantly improves learning efficiency and achieves higher performance than reinforcement learning (Zhu et al., 2017; Hester et al., 2017). However, it is expensive and time-consuming to collect diverse human demonstrations with high quality. We believe that virtual reality games can provide us with an ideal medium to crowdsource demonstrations from a broad range of users (von Ahn and Dabbish, 2008).

In this work, we focus on simulating cooking activities in a virtual kitchen environment, `VRKitchen`. We illustrate how this system can address the emerging needs for the learning problems in an example shown in Figure 3.4, where an agent makes a sandwich in one of the kitchens created in our system.

- The environment allows the agent to interact with different *tools* and *ingredients* and simulates a variety of object changes; *e.g.*, the bread changes its color when it is being heated in the oven, and the tomato turns into slices after it is cut. The

agent's interactions with the physical world when performing cooking tasks will result in large variations and temporal changes in objects' appearance and physical properties, which calls for a task-oriented visual representation.

- To make a sandwich, the agent needs to perform a long sequence of actions, including taking ingredients from a fridge, putting cheese and ham on the bread, toasting the bread, adding some sliced tomato, and putting some sauce on the bread. To quickly and successfully reach the final goal, it is necessary to equip the agent with the ability to conduct long-term planning.
- We build two interfaces to allow an AI algorithm as well as a human user to control the embodied agent respectively; thus, humans can give demonstrations using VR devices at any place in the world, and the AI algorithms can learn from these demonstrations and perform the same tasks in the same virtual environments.

In summary, our main contributions are:

- A configurable virtual kitchen environment in a photo-realistic 3D physical simulation which enables a wide range of cooking tasks with rich object state changes and compositional goals;
- A toolkit including a VR-based user interface for collecting human demonstrations and a Python API for training and testing different AI algorithms in the virtual environments.
- A new human demonstration dataset of various cooking tasks – UCLA VR chef dataset.
- A multi-view dataset automatically generated from `VRKitchen` with automatically generated annotations.

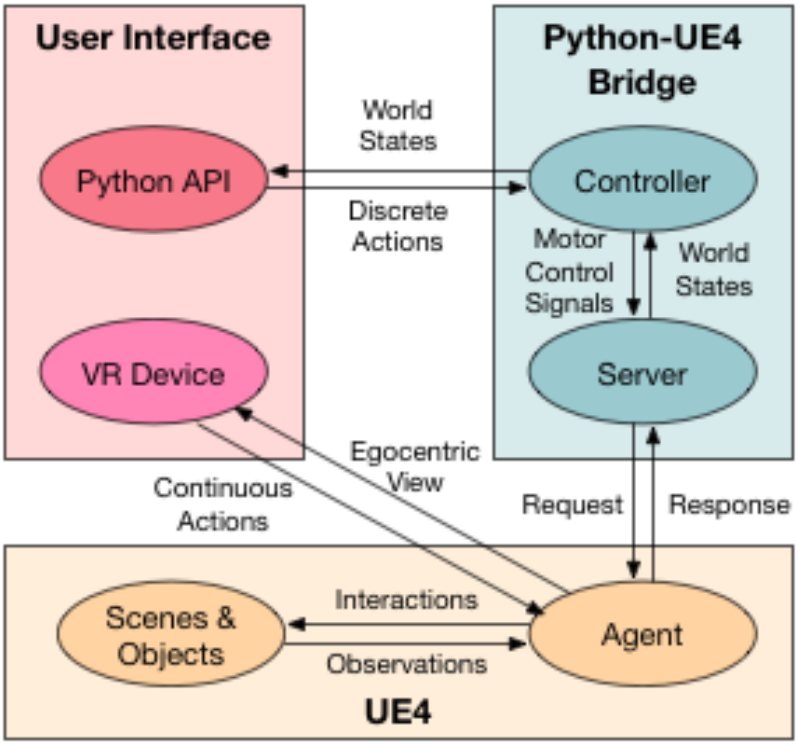


Figure 3.5: Architecture of `VRKitchen`. Users can either directly teleoperate the agent using VR device or send commands to the agent by Python API.

3.2 The `VRKitchen` Environment

Our goal is to enable better learning of autonomous agents for tasks with compositional goals and rich object state changes. To this end, we have designed `VRKitchen`, an interactive virtual kitchen environment which provides a testbed for training and evaluating various learning and planning algorithms in a variety of cooking tasks. With the help of virtual reality device, human users serve as teachers for the agents by providing demonstrations in the virtual environment.

3.2.1 Architecture Overview

Figure 3.5 gives an overview of the architecture of `VRKitchen`. In particular, our system consists of three modules: (1) the physics engine and photo-realistic rendering module consist of several humanoid agents and kitchen scenes, each has a number of ingredients

Figure 3.6: Four humanoid avatars designed using MakeHuman.

and tools necessary for performing cooking activities; (2) a user interface module which allows users or algorithms to perform tasks by virtual reality device or Python API; (3) a Python-UE4 bridge, which transfers high-level commands to motor control signals and sends them to the agent.

3.2.2 Physics Engine and Photo-realistic Rendering

As a popular game engine, Unreal Engine 4 (UE4) provides physics simulation and photo-realistic rendering, which are vital for creating a realistic environment. On top of that, we design humanoid agents, scenes, object state changes, and fine-grained actions as follows.

3.2.3 Humanoid Agents

Agents in `VRKitchen` have human-like appearances (shown in Figure 3.6) and detailed embodiment representations. The animation of the agent can be broken into different states; *e.g.walking, idle*. Each agent is surrounded by a capsule for collision detection: when it's *walking*, it would fail to navigate to a new location if it collides with any objects in the scene. When it is *idle*, the agent can freely interact with objects within a certain range of its body.

There are, in total, 12 different animation states as shown in Figure 3.7. Each animation state has an associated animation. The transitions of the animation states are determined by python API data. When appropriate, the python API will issue an

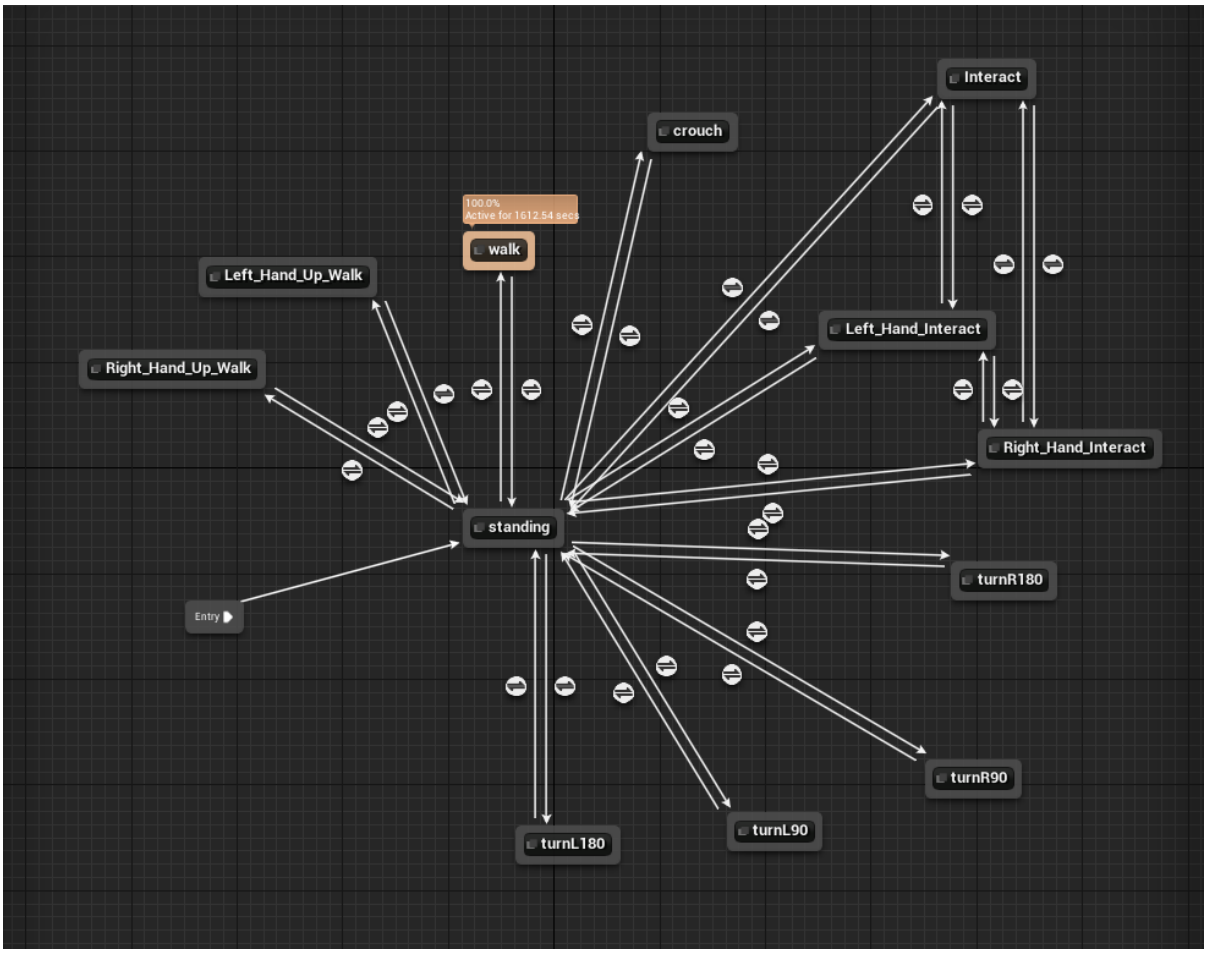


Figure 3.7: Animation states for our agents

animation transition command. There are two types of animations: i) node manipulation through inverse kinematics(IK) ii) blended animations from different online resources. IK systems are more flexible; however, the final animation might not look natural at all. This is because the IK system will try to reach the location of the specification regardless of the pose of the character. Blended animations, on the other hand, are more natural but can not reach any arbitrary position in the environment. Therefore, for animation states that require object manipulations or related to object manipulations, reaching for an object, or crouching down, we are using the IK system. For animation states like holding objects and walking, turning around, and standing up, we are using blended animations. The transition between animation states is blended using the tool provided by UE4. We chose the Hermite cubic transition mode with a transition duration between 0.2 seconds and 0.5

Figure 3.8: VRKitchen scenes

seconds based on different transitions. We manually tried different modes and transition duration and found out this parameter setting looks more natural. Even though all agents share the same animation states, different agents may exhibit different behaviors when using IK animations. This is because different agents have different limb lengths, so the computed IK trajectory might be different.

3.2.4 Scenes

`VRKitchen` consists of 16 fully interactive kitchen scenes as shown in Figure 3.8. Agents can interact with most of the objects in the scenes, including various kinds of *tools, receptacles* and *ingredients*. Each kitchen is designed and created manually based on a common household setting. 3D models of furniture and appliances in kitchens are first obtained from the SUNCG dataset (Song et al., 2017). SUNCG dataset provides a script to create an entire kitchen from different 3D models. However, the created kitchens do not support any interactions at all. For example, agents cannot open the doors in the kitchen (stove door, cabinet doors, *etc.*.), because doors are fixed, not movable. To solve this issue, we use blender to manually separate door from the rest of the 3D model for various different 3D models as shown in Figure 3.9. We also decompose other parts of the

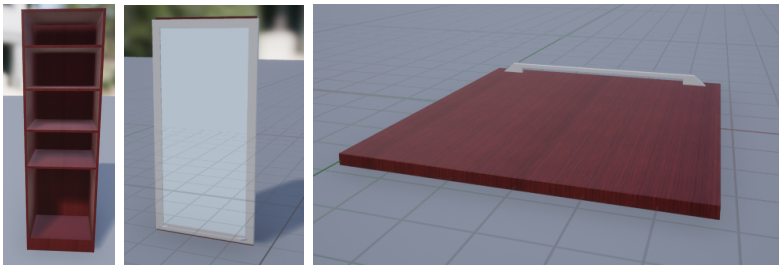


Figure 3.9: Sample decomposed kitchen cabinet. Manually decomposed through blender.

objects according to our need. Sometimes decomposing objects are not enough. A decent amount of SUNCG models do not have interiors at all. In order to make it functional in our kitchen setting, we also manually design and create the functional interiors for our 3D models using blender. After we have basic furniture and appliances in the scene, we then add cooking *ingredients* and *tools*. Instead of sampling their locations randomly, we place the objects according to their utility; *e.g.*, *tools* are placed on the cabinets while perishable *ingredients* such as fruits and vegetables are available in the fridge. On average, there are 55 interactive objects in a scene.

3.2.5 Object State Changes

One key factor of `VRKitchen` is the ability to simulate state changes for objects. Instead of showing only pre-conditions and post-effects of actions, `VRKitchen` simulates the continuous geometric and topological changes of objects caused by actions. This leads to a great number of available cooking activities, such as roasting, peeling, scooping, pouring, blending, juicing, *etc.*. Overall, there are 18 cooking activities available in `VRKitchen`.

The environment mainly consists of discrete changes as shown in Figure 3.10a and Figure 3.10b. We believe that, for most tasks, discrete fluent changes that specify pre-conditions and post-effects are sufficient for task planning.

(a) The inset shows a tomato before a cutting action

(b) The inset shows a tomato after a cutting action

Figure 3.10: Before and after a cutting action

3.2.6 Fine-Grained Actions

In previous platforms (Kolve et al., 2017; Brodeur et al., 2017), objects are typically treated as a whole. However, in the real world, humans apply different actions to different parts of objects; *e.g.*, to get some coffee from a coffee machine, a human may first press the power button to open the machine, and press the brew button afterwards to brew coffee. Thus we design the objects in our system in a compositional way; *i.e.*, an object has multiple components, each of which has its own affordance. This extends the typical action space in prior systems to a much larger set of fine-grained actions and enables the agents to learn object-related causality and commonsense.

3.2.7 User Interface

With a detailed human embodiment representation, multiple levels of human-object interactions are available. In particular, there are two ways for users to provide such demonstrations:

(1) Users can directly control the agent's head and hands. During teleportation, actions are recorded using a set of off-the-shelf VR devices, in our case, an Oculus Rift head-mounted display (HMD) and a pair of Oculus Touch controllers. Two Oculus constellation sensors are used to track the transforms of the headset and controllers in 3D spaces. We then apply the data to a human avatar in the virtual environment: the avatar's head and hand movements correspond to the human user's, while other parts of its body are animated through a built-in Inverse Kinematics solver (Forward And Backward Reaching Inverse Kinematics, or FABRIK). Human users are free to navigate the space using the Thumbsticks and grab objects using the Trigger button on the controller. Figure 1.1 gives an example of collecting demonstrations for continuous actions.

(2) The Python API offers a way to obtain discrete action sequences from users. In particular, it provides world states and receives discrete action sequences. The world state is comprised of the locations and current states of nearby objects and an RGB/depth image of the agent's first-person view. Figure 3.11 and Figure 3.12 show examples of

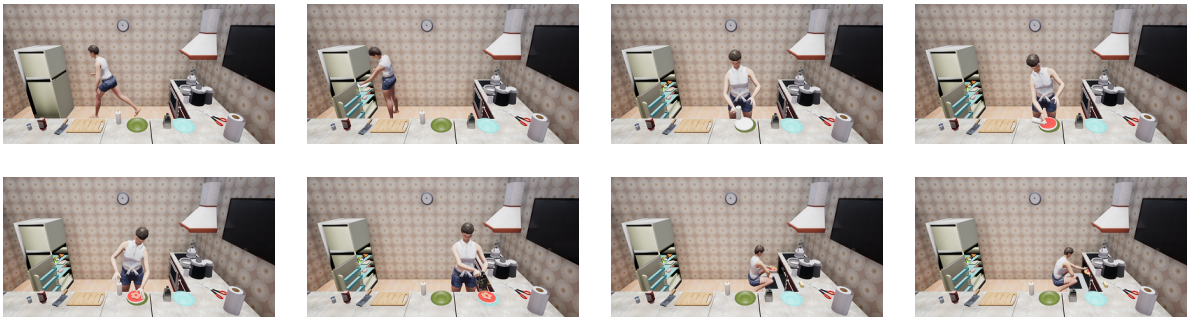


Figure 3.11: An example of human demonstrations for making a *pizza*

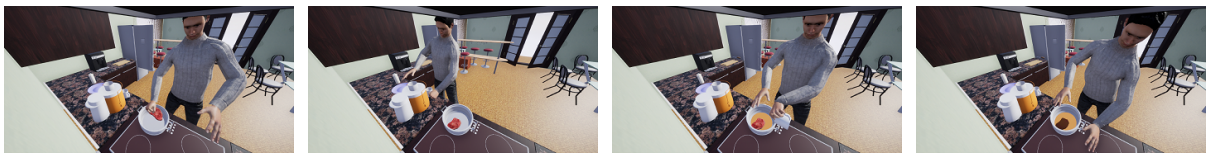


Figure 3.12: An example of human demonstrations for making *roast meat*

recorded human demonstrations for tasks *pizza* and *roast meat* from a third person view.

3.3 Python-UE4 Bridge

The Python-UE4 bridge contains a communication module and a controller. The Python server communicates with the game engine to receive data from the environment and send requests to the agent. It is connected to the engine through sockets. To perform an action, the server sends a command to UE4 and waits for a response. A client in the game engine parses the command and applies the corresponding animations to the agent. A payload containing states of nearby objects, the agent's first-person camera view (in terms of RGB, depth, and object instance segmentations), and other task-relevant information are sent back to the Python server. The process repeats until the terminal state is reached.

The controller enables both low level motor controls and high level commands. Low-level controls change local translation and rotation of the agent's body, heads, and hands, while other body parts are animated using FABRIK. High-level commands, which perform atomic actions such as taking or placing an object, are further implemented by taking

advantage of the low-level controller. To cut a carrot with a knife, for example, the high-level controller iteratively updates the hand location until the knife reaches the carrot.

3.4 Performance

We run `VRKitchen` on a computer with Intel(R) Core(TM) i7-7700K processor @ 4.50GHz and NVIDIA Titan X (Pascal) graphics card. A typical interaction, including sending a command, executing the action, rendering the frame, and getting a response, takes about 0.066 seconds (15 actions per second) for a single thread. The resolutions for RGB, depth, and object segmentation images are by default 84×84, but can be changed to any resolution if needed(will affect performance).

3.5 Environment Interactions

In `VRKitchen`, we design all atomic actions and object state changes available in several dish preparation tasks. Using these atomic actions, the agent can interact with the environments until a predefined goal is reached. Figure 3.13 shows some examples of dishes.

3.5.1 Atomic Actions

Each atomic action listed below can be viewed as a composition of a verb (action) and a noun (object). Objects can be grouped into three types: *tools, ingredients, and receptacles*. (1) *Ingredients* are small objects needed to make a certain dish. We assume that the agent can hold at most one *ingredient* at a time. (2) For *receptacles*, we follow the definition in (Kolve et al., 2017). They are defined as stationary objects which can hold things. Certain *receptacles* are called *containers*, which can be closed, and agents can not interact with the objects within them until they are open. (3) *Tools* can be used to change the states of certain *ingredients*. Atomic actions and object affordance are defined in the

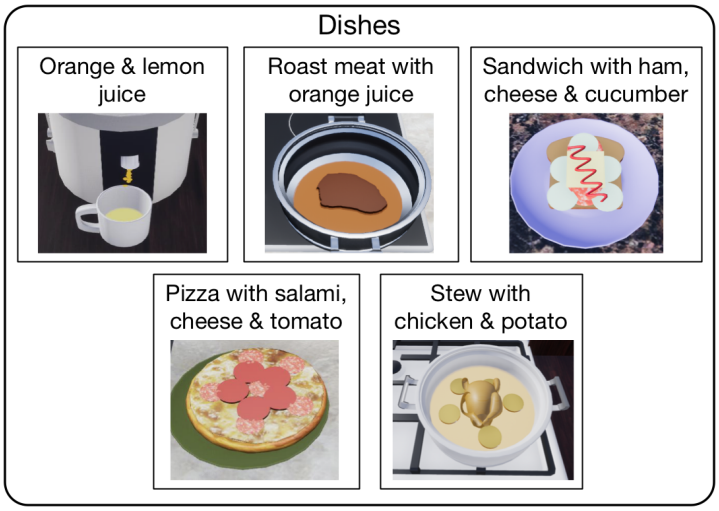


Figure 3.13: Examples of dishes made in `VRKitchen`. Note that different *ingredients* leads to different variants of a dish. For example, mixing orange and kiwi juice together would make *orange & kiwi juice*.

following way:

- `Take` {*ingredient*}: take an *ingredient* from a nearby *receptacle*;
- `Place into` {*receptacle*}: put a held *ingredient* into a nearby *receptacle*;
- `Use` {*tool*}: use a *tool* to change the state of a *ingredient* in a nearby *receptacle*;
- `Go To` {*tool, receptacle*}: move to a *tool* or *receptacle*;
- `Toggle (open/close)` {*container*}: change state of a *container* in front of the agent.
- `Turn`: rotating the agent's facing direction by 90 degrees.

Note that actions including `Take, Place into, Use`, and `Toggle` would fail if the agent is not near the target object.

3.5.2 *Ingredient* Sets and States

Meanwhile, there are seven sets of *ingredients*, including *fruit, meat, vegetable, cold-cut, cheese, sauce, bread* and *dough*. Each set contains a number of *ingredients* as variants: for example, *cold-cut* can be ham, turkey, or salami. One *ingredient* may have up to four

Task	Goal states	Target location
Fruit juice	fruit1: cut, juiced; fruit2: cut, juiced	cup
Roast meat	fruit: cut, juiced, cooked; meat: cooked	pot
Stew	veg: cut, cooked; meat: cooked	pot
Pizza	veg: cut, cooked; cold-cut: cooked; cheese: cooked; sauce: cooked; dough: cooked	plate
Sandwich	veg: cut; sauce; cold-cut: cooked; cheese: cooked; bread: cooked	plate

Table 3.1: The goals for five available dishes. In each task, the agent should change required *ingredients* to the goal states and move them to a target location.

types of state changes: *cut, peeled, cooked* and *juiced*. We manually define affordance for each set of *ingredients*: *e.g.*, *fruit* and *vegetable* like oranges and tomatoes can be juiced (using a juicer) while bread and meat can not. *Tools* include grater, juicer, knife, oven, sauce bottle, stove, and *receptacles* are fridge, plate, cut-board, pot, and cup.

3.5.3 Goals

Based on the atomic actions defined in Section 3.5.1, agents can prepare five dishes: *fruit juice, stew, roast meat, sandwich* and *pizza*. The goals of each task are compositionally defined upon (1) goal states of several sets of ingredients and (2) target locations: to fulfill a task, all required *ingredients* should meet the goal states and be placed in a target location. For example, to fulfill the task *fruit juice*, two *fruits* should be cut, juiced, and `placed into` the same cup. Here, the target locations are one or several kinds of *containers*. Table 3.1 defines the goal states and target locations of all tasks.

Figure 3.14: Using a VR device at home

3.6 Data Generation From Virtual Environment

3.6.1 Gather Data From Human Demonstrations

Gathering Human demonstrations is an essential component of policy-learning tasks. DAGGER (Ross et al., 2010) requires humans to continuously provide feedback to the learned policy and take over when a human sees a mismatch between agents' policy and human belief. A distributed data collection tool also provides future opportunities for crowd-sourcing human demonstrations. Here, we provide two different ways for humans to take over when appropriate: 1) through a VR device and 2) through a web-based interface in the case of not having a VR device available.

3.6.2 Gather Human Demonstration From VR Device

UE4 has built-in VR support. Here, we use Oculus Rift to perform our experiment. In the VR Setting, a lot of actions are continuous; however, in VR Kitchen, atomic actions are discrete. In order to mitigate this difference, we propose to decompose continuous actions into discrete ones. Users use Touch Pad "A" Button and pointers to navigate around the world to make appropriate actions. As long as users have VR device, they are not constraint on their locations as shown by Figure 3.14 and Figure 3.15.

Figure 3.15: Using a VR device in the office

Figure 3.16: Web interface instructions

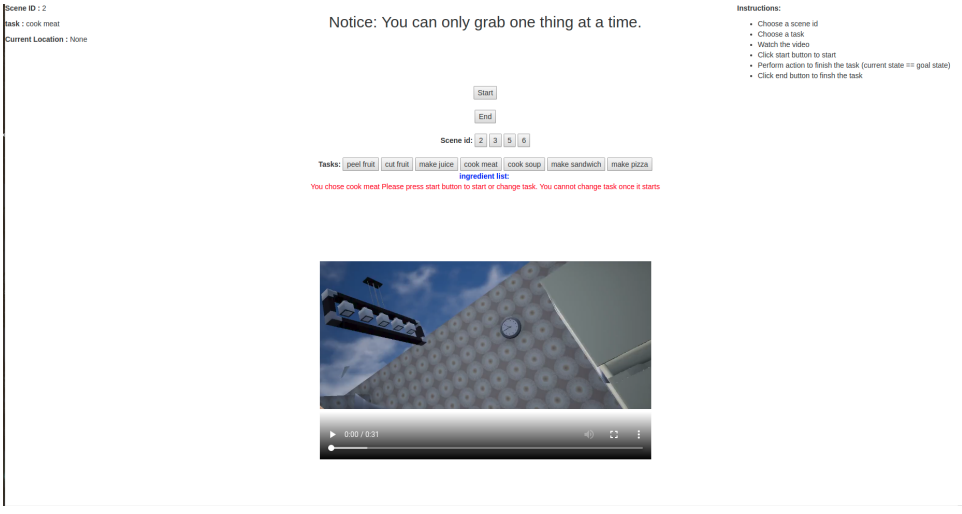


Figure 3.17: Web interface tutorial

i) When the user presses the "A" button, provided the user is far away from the location of interest, the system will teleport the user to the nearest valid location around his pointer. In the back end, the system will interpret this user action as GoTo *location*.

ii) When the user presses the "A" button, provided the user is close to the location of interest, the system will interpret the action as one of the following according to the current state of the agent: Use *item*, Take *item*, Place into *item*, Toggle(Open/Close) *item*.

3.6.3 Gather Human Demonstration From Web Interface

VR devices are still quite expensive to this date. Therefore, not everyone has a VR device available in their home. In order to alleviate this problem, we propose to use a web-based interface for the purpose of data collection for the general public. The web-based interface is built using Flask, JavaScript, HTML5, and CSS.

At first, the web-based interface will provide some initial text-based instructions about the task as shown in Figure 3.16, and the users are asked to solve this problem with their commonsense knowledge.

After users choose a task and a scene ID, the web-based interface will provide a short and quick demo of the environment set-up and sample demonstrations done by the machine for that task. As shown by Figure 3.17. However, the ingredients are randomized. Therefore, users are most likely to use a different set of ingredients.

Then the web-based interface will display all the valid atomic actions to the users and provide users with a text-based description about the goal state of the current task and the user's current state as shown in Figure 3.18. Users can simply click buttons on a web browser to execute an action. All actions are recorded in the back end. The actions and the associated user RGB image can be used for imitation learning.

In the data collection process, users will first solve a simple task "Cut Fruit", which is not recorded so that users are familiar with the tools and the environment.

We recorded the human demonstrations from 9 different users, and we found out that

Figure 3.18: Web interface atomic actions

Task	Average (Number of steps to solve the task)	Median	Variance
Cook Meat	16.67	16	7.8
Cook Soup (only 8 data points are valid)	15.38	14.5	9.9
Make Juice	17.67	17	6.9
Make Pizza	34.11	33	28.8
Make Sandwich	30.67	29	12.67

Table 3.2: Human demonstration statistics

the user's background has a significant impact on task completion steps. For example, users from a Western background can finish the "Make Pizza" task way faster than others. We also found out that simple tasks tend to have a lower variance. Detail statistics are shown in Table 3.2

3.7 Generate Data From Ground Truth

In General Computer vision tasks, we often need to annotate data. With our environment, we can automatically provide some annotated data with low noise. We can provide RGB images, depth images, and instance segmentation from multi-view cameras. We also has the capacity to automatically generate bounding boxes on a 2D image. Apart from those general computer vision annotations, we can also provide annotations tailored to a specific computer vision task.

Here, we demonstrate our environment's capacity through an AttentionObject-VR dataset.

3.7.1 Human Attention

Attention is an important topic in the computer vision field and has been widely used for object detection, video tracking, image retrieval, and other applications. Eye fixation saliency map estimation and saliency object estimation are two important problems in the study of visual attention, and their focus is inferring saliency regions or objects in an image that draw the attention of the human (outside the image) who is looking at the image. In this chapter, we study the attention of a human inside a third-person view video, we call it Inside-video human attention. To infer human attention, the foremost thing is to make clear what human attention is. Originally, attention is a concept in philosophy. Nowadays, it is well-known as a concept in psychology. One dominant definition in psychology is that attention is the process of attending to objects. This definition indicates that attention is based on objects. Actually, some studies (Chen, 2012; Chou and Yeh, 2012; Pooresmaeili and Roelfsema, 2014) in psychophysics and biology fields as well as some inter-discipline studies in neuro image filed and brain image field also claim the object-based attention. These studies provide the strong theory support for defining human attention as objects. Another widely accepted definition in psychology is that attention is something that happens in the mind—a mental "inside" which is linked with the perceivable "outside" (Seemann, 2011). This definition indicates that attention is related to the high-level invisible information in the human mind.

Based on these studies, we define human attention as the attentional objects that coincide with the task a human is doing. With a task in mind, a human finishes the task by doing several sub-tasks in a certain temporal order. For example, when a human is doing the task of taking the water from the drinking fountain, the human first finds the cup, then goes to the drinking fountain, and finally takes the water. To finish each sub-task, a human behaves purposely to operate on or approach the attentional objects.

For example, when the human is doing the sub-task of finding the cup, the human uses the hand to catch the cup. When the human is doing the sub-task of going to the drinking fountain, the human walks to approach the drinking fountain.

3.7.2 Dataset Overview

Though there exists a large number of datasets for the studies of human gaze, visual attention, and human-object interaction, to our best knowledge, no publicly available dataset is targeted for inferring the task-driven inside video human attention. Therefore, we collect a video dataset in VR (Virtual Reality) scenes. With the development of VR techniques, VR data has become as life-like as real data. In VR scenes, all objects are configured with accurate locations and sizes, allowing automatic object annotations and large-scale data collection. To collect the dataset, we use 8 different existing kitchen scenes. In each scene, many furniture and objects are configured, objects can be divided into two categories: tools (*e.g.*, knife, juicer, oven, *etc.*.) and ingredients (*e.g.*, bread, orange, tomato, *etc.*.). A human can use tools to change the state of an ingredient. For example, to make orange juice, a human uses a knife to cut an orange into halves and put them into a juicer to get juice.

The dataset has several characteristics:

- Diverse and large. The dataset consists of 8 scenes, 10 tasks, 33 subtasks, and 4 humans. As shown in Figure 3.19, different scenes vary significantly in the scene scale, furniture configuration, and object placement. For each scene, we collect videos from 3 different camera views to make the data more diverse. The camera views are fixed in certain scene, and are manually chosen to make the views cover the 360-degree scene. The images of different camera views notably differ from each other. The 10 tasks are: bake bread, cook soup, cut meat, fry steak, make coffee, make juice, make sandwiches, microwave food, pour coke, and turn on the light. The dataset is large, consisting of 133,419 images and 1,887,858 object annotations in total. The detailed statistics are in Table 3.3. On average, each video consists of

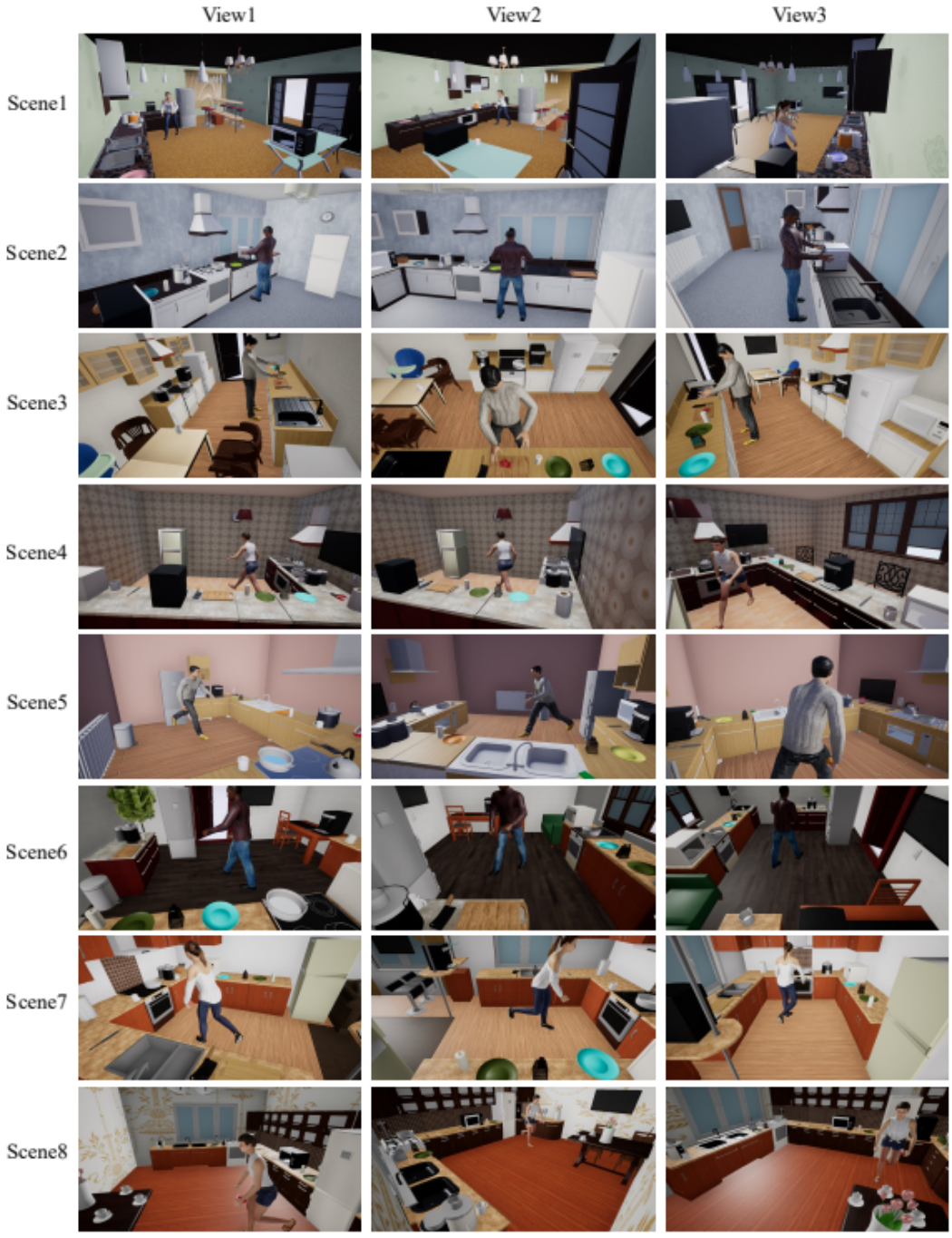


Figure 3.19: Samples of the AttentionObject-VR dataset. The dataset is collected in eight scenes. In each scene, videos are captured from three different camera views. In this figure, each row shows three images from the three camera views at the same time in the same scene

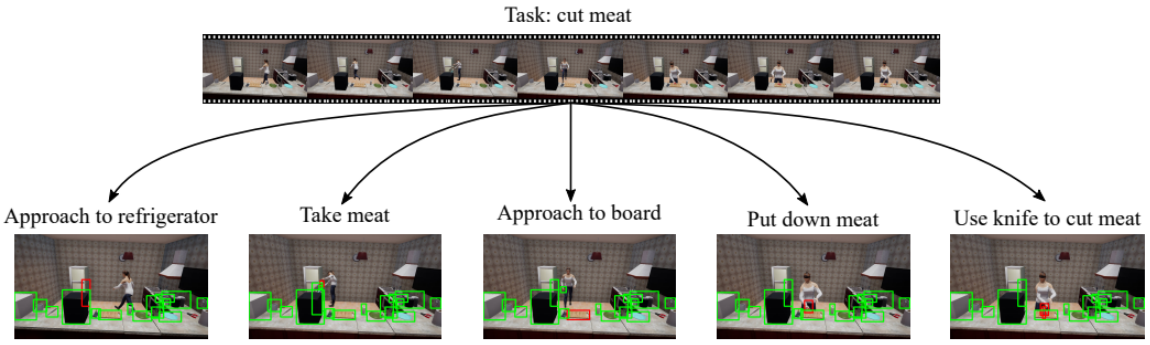


Figure 3.20: An example of annotating a video. Given a video with the task label of "cut meat", the video is segmented into several sub-tasks ("approach to refrigerator", "take meat", "approach to board", "put down meat", and "use a knife to cut meat"). In each sub-task, the attentional object (red bounding boxes) and other non-attentional objects (green bounding boxes) are annotated. To conclude, the annotations include task labels, sub-task labels, attentional objects, and non-attentional objects.

	Videos	Images	Attentional Objects	Other Objects
Train	596	100,951	117,643	1,330,431
Test	184	32,468	37,211	402,573
Total	780	133,419	154,854	1,733,004

Table 3.3: The statistics of the AttentionObject-VR dataset. Videos: Number of videos, Images: Number of images, Attentional objects: number of attentional object annotations, other objects: Number of non-attention object annotations

171 images. The video resolution resolution is 1280 × 720.

- Well-organized. To make the dataset qualified for inferring human attentional objects, it is necessary to guarantee humans and attentional objects are inside images. Therefore, we removed the images and videos that did not satisfy this requirement. To divide the dataset into training sets and testing sets, the data collected in scenes 7 and scene 8 are used for testing, and the data collected in other scenes is used for training. Table 3.4, Table 3.5 and Table 3.6 show some of the sample tasks.

- Well-annotated. Figure 3.20 shows an example of annotating a video. Given a video with a task label, it is segmented as several sub-tasks to guarantee that

Task	Task definition (attention objects in parenthesis)
Make coffee	Approach cup (cup) Take cup (cup) Approach coffee machine (coffee machine) Put the cup under coffee machine (cup and coffee machine) Press make button to make coffee (cup and coffee machine)
Microwave Food	Approach fridge (fridge) Open door of fridge (fridge) Take the bread (bread) Close fridge (fridge) Approach plate (plate) Put bread on plate (bread and plate) Take the plate and bread to approach microwave (microwave) Use microwave (microwave)
Cook Soup	Approach tomato (tomato) Use knife (knife and tomato) Pick up tomato (tomato) Approach pot (pot) Put the tomato into pot (pot and tomato)
Pour Coke	Approach fridge (fridge) Take coke from fridge (coke) Close fridge (fridge) Approach a cup (cup) Pour coke into the cup (cup and coke)

Table 3.4: Dataset Tasks (Part 1)

the attentional object in each sub-task is determinate. To accurately segment a task into several sub-tasks, three volunteers are asked to find the keyframes in a video to segment sub-tasks. In most cases, the keyframe is not controversial. The average key frame is taken as the final key frame for controversial ones. For each frame, the attentional objects and non-attentional objects are annotated with detailed information like the location, size, and types. On average, each image contains 1.16 attentional object annotations (two attentional objects are annotated in some images) and 13 non-attentional object annotations. Benefiting from the

Task	Task definition (attention objects in parenthesis)
Make Juice	Approach fridge (fridge) Open fridge (fridge) Take up orange (orange) Close fridge (fridge) Approach board (board) Use knife (knife and orange) Take the orange (orange) Approach juicer (juicer) Put the orange into juicer (orange and juicer)
Fry Steak	Approach fridge (fridge) Open door of fridge (fridge) Take the steak(bread) Close fridge(fridge) Approach pot (pot) Put the steak into pot (pot and steak) Operate stove (stove)
Make Sandwich	Approach fridge (fridge) Open door of fridge (fridge) Take the bread (bread) Close fridge (fridge) Approach plate (plate) Put the bread onto plate (bread and plate) Approach fridge (fridge) Open door of fridge (fridge) Take the ham (ham) Close fridge (fridge) Approach bread (bread) Put the ham onto bread (ham and bread)

Table 3.5: Dataset Tasks (Part 2)

Task	Task definition (attention objects in parenthesis)
Bake Bread	Approach fridge (fridge) Open door of fridge (fridge) Take the bread (bread) Close fridge (fridge) Approach oven (stove) Use oven (stove, bread)
Cut Meat	Approach fridge (fridge) Open fridge door (fridge) take the beef (beef) Close fridge door (fridge) Approach board (board) use knife (knife, beef)
Turn on the light	Approach light switch (light switch) push/pull switch (light switch)

Table 3.6: Dataset Tasks (Part 3)

good annotations, the dataset can also be used for other studies like task/event recognition, video segmentation, and action recognition.

3.7.3 Dataset Benchmark Results

We run object detection algorithm RetinaNet (Lin et al., 2017) on this dataset, and the results are in Table 3.7. Then we benchmark our dataset on human attention task.

We study the problem of inferring the task-driven attentional objects of a human inside third-person view videos, to our best knowledge, there does not exist exactly same work with ours. The most related work is to estimate where a human is looking. Therefore, we select two state-of-the-art human face and head direction estimation methods as baselines. We briefly describe the three baseline methods as follows:

- PRNet (Feng et al., 2018) is a face alignment method that can estimate human face direction. It takes the raw image and human face as input, and the output is the dense (more than 40K) aligned face key points. These dense points are compared with a pretrained model to compute the camera matrix, which is further combined

Class	Num. of Instances	Accuracy (mAp)
Bread	4409	0.1665
Cut Board	29828	0.0340
Microwave	26144	0.4670
Fridge	28912	0.7911
Light Switch	8146	0.4900
Coke	3014	0.2539
Stove	28640	0.6707
Juicer	30402	0.9961
Coffee Machine	21487	0.0351
Plate	57606	0.7812
Ham	838	0.0021
Beef	2602	0.0987
Tomato	1780	0.1883
Cup	61534	0.5512
Pot	30678	0.6622
Eggplant	1129	0.0153
Knife	29488	0.2207
Orange	1417	0.1404
Average by Class		0.3647
Weighted Average		0.5390

Table 3.7: Object detection results

with 68 facial key points to estimate the human face direction.

- Hopenet (Ruiz et al., 2018) is a head pose estimation method. It takes the raw image and human face as input, and the output is the three Euler angles that signal the human head direction.

- ResNet-BinCls (He et al., 2016) is a binary classification method based on ResNet-18 (He et al., 2016). It first detects the objects in an image; then, a binary classifier estimates the scores of each object being and not being the attentional object. To estimate the score of a candidate object, the human skeleton and the candidate object are represented as a binary $1 \times H \times W$ mask, which is concatenated with the $3 \times H \times W$ raw image to serve as the input of the trained binary classifier. The RetinaNet model (Lin et al., 2017) and OpenPose model (Cao et al., 2017) are respectively used for attentional object candidate detection and human pose

Methods	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	All
PRNet	0.41	0.28	0.29	0.28	0.26	0.29	0.31	0.34	0.27	0.07	0.30
Hopnet	0.54	0.36	0.36	0.37	0.17	0.37	0.39	0.39	0.29	0.00	0.35
ResNet-BinCls	0.49	0.51	0.46	0.55	0.19	0.53	0.48	0.71	0.50	0.48	0.48

Table 3.8: Accuracy of different methods on the AttentionObject-VR dataset. "All" corresponds to the overall accuracy. T1 to T10 correspond to the accuracy of different tasks. T1: bake bread, T2: cook soup, T3: cut meat, T4: fry steak, T5: make coffee, T6: make juice, T7: make sandwich, T8: microwave food, T9: pour coke, and T10: turn on light.

estimation.

Benchmark results are shown in Table 3.8, and qualitative results are shown in Figure 3.21.

3.8 Summary

We have designed a virtual reality system, `VRKitchen`, which offers physical simulation, photo-realistic rendering of multiple kitchen environments, a large set of fine-grained object manipulations, and embodied agents with human-like appearances and gestures. We have implemented toolkits for training and testing AI agents as well as for collecting human demonstrations in our system. We were also able to compile a video dataset of human demonstrations of the cooking tasks using the user interface and scripting files in the system.

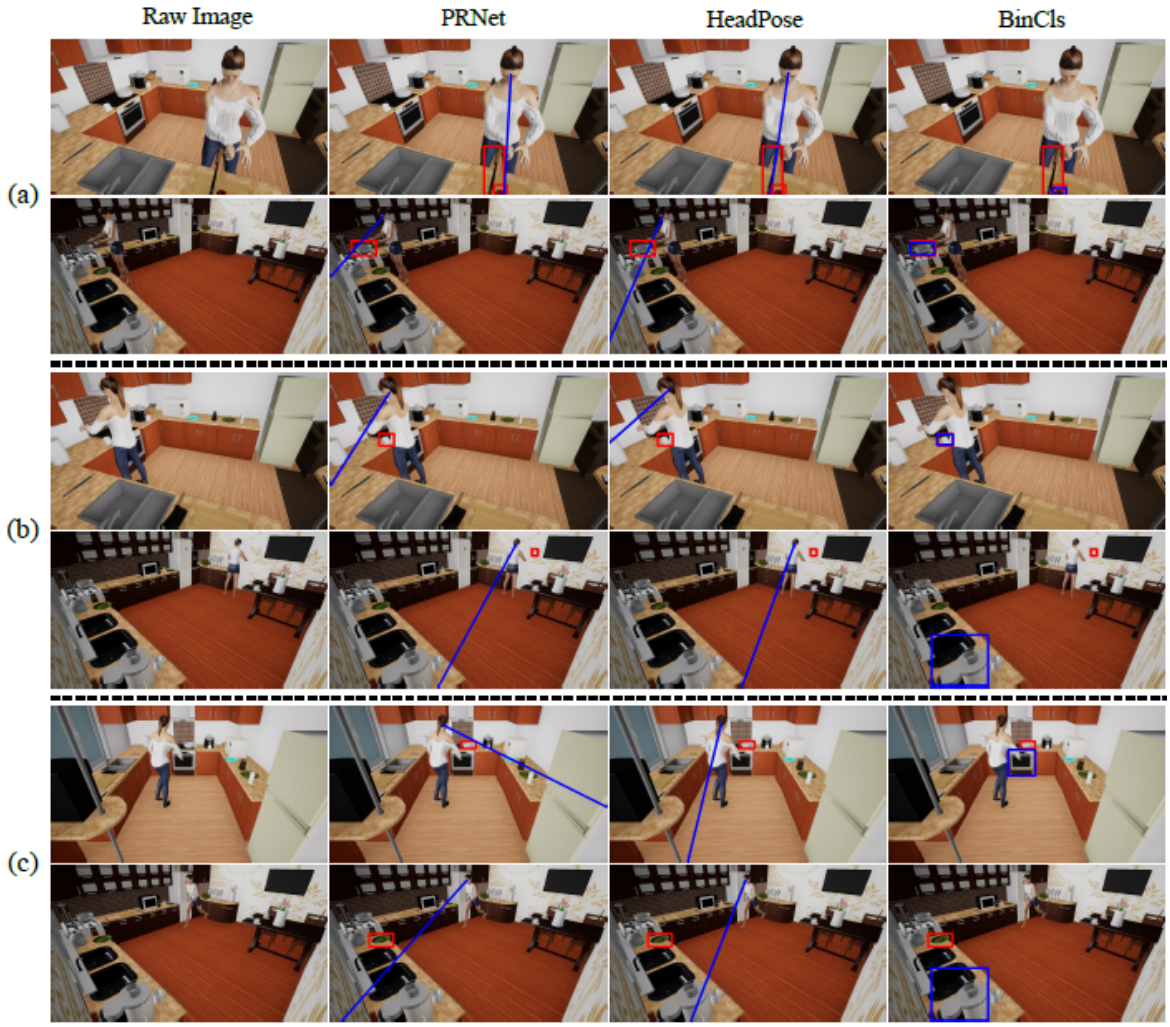


Figure 3.21: Samples of qualitative results of different methods in three typical scenarios. (a) Human facial information is available and conveys a distinct cue to infer attentional objects. (b) Human facial information is not available, but the human pose provides the informative cue to infer attentional objects. (c) Human facial cues and human pose cues are not sufficient, and invisible high-level task information is needed to infer attentional objects. In this Figure, the red bounding boxes represent the ground truth attentional object annotations, the blue lines represent the face and head directions estimated by the PRNet model and Hopenet model, and the blue bounding boxes represent the attentional objects estimated by the ResNet-BinCls method.

CHAPTER 4

ARNOLD: A Benchmark for Language-Grounded Task Learning With Continuous States in Realistic 3D Scenes

4.1 Introduction

The ability to ground language is a crucial skill that has evolved over the course of human history, allowing people to learn and describe concepts, perform tasks, and communicate with one another. While recent developments have enabled the grounding of concepts in images (Radford et al., 2021; Kamath et al., 2021; Saharia et al., 2022; Gao et al., 2016), the interaction with physical environments (Deitke et al., 2020; Batra et al., 2020; Xia et al., 2020; Li et al., 2021; Nagarajan and Grauman, 2020; Jain et al., 2019, 2020; Gu et al., 2022; Geng et al., 2022, 2023; Xu et al., 2023; Wan et al., 2023; Deitke et al., 2022a; Duan et al., 2022), and language understanding in physical environments (Lu et al., 2019; Anderson et al., 2018; Pashevich et al., 2021; Zhang and Chai, 2021; Thomason et al., 2020; Gao et al., 2022; Wang et al., 2019; Padmakumar et al., 2022), few researchers have investigated the grounding of actions in daily tasks (Shridhar et al., 2022a; Zheng et al., 2022; Shridhar et al., 2022b; Wang et al., 2022). Given that humans can comprehend object status and relate language instructions to the physical environment, a pertinent question arises: *How can we imbue robotic systems with the same capacity to understand and execute language instructions in the physical world?*

Learning action grounding in daily activities is a challenging task that presents several non-trivial difficulties. Firstly, robotic tasks rely heavily on detailed scene understanding for successful execution. This includes the understanding of geometry information, layouts, and visual appearances. The various combinations of scene configurations, including novel

appearances, objects, and spatial positions, further exacerbate this challenge. Therefore, it is crucial for robotic systems to acquire generalizable skills that can be transferred to different domains and settings.

Secondly, humans possess an exquisite ability to understand desired goal states precisely. This ability allows us to effortlessly map from simple descriptions (*e.g.*, a cup half filled, a door fully opened, *etc.*) to the precise status of physical properties (*e.g.*, half the volume, pulled to 180°, *etc.*). However, it is exceedingly challenging for robots to learn the precise goal state from abstracted language instructions, especially when referring to an implicit range of continuous object states (*e.g.*, a bit of coffee, slightly open, *etc.*) (Krantz et al., 2020; Hong et al., 2022; Mees et al., 2022a). As a result, there is an urgent need for robot systems to maintain a mapping from language instructions to precise goal states in a continuous world.

A necessary first step toward tackling these challenges is to develop realistic robot simulation systems that enable language-grounded learning. Indeed, notable recent advances in simulated environments have facilitated grounded task learning (Das et al., 2018; Shridhar et al., 2020a; Mees et al., 2022b; Zheng et al., 2022; Ma et al., 2022). Despite the impressive progress, these benchmarks suffer from several limitations that hinder the effective operation of robots in the real world: (1) They typically assume that tasks are performed in simple and clean environments rather than in scenes that are spatially occupied by clutter and visually disturbed by diverse textured backgrounds (Kumar and Todorov, 2015; Lin et al., 2020; Zheng et al., 2022; Shridhar et al., 2022b). (2) They assume discrete (*e.g.*, binary) object states and perfect motor control that ignore the low-level geometry and dynamics of objects (Szot et al., 2021; Srivastava et al., 2022; Ehsani et al., 2021) and, consequently, they do not attempt in-depth physical state understanding or fine-grained manipulation skills. (3) These benchmarks do not ground instructions to precise states (Zheng et al., 2022; Shridhar et al., 2022a), thus neglecting the challenging problem of grounding language to object states.

To address these critical challenges of language-grounded robot task learning, we introduce a new benchmark, `ARNOLD`, for grounding task instructions to precise robot

actions and object states in realistic natural scenes (Figure 1.1). Specifically, we leverage a highly accurate physics simulation engine to create eight challenging robot manipulation tasks that include continuous robot motion, friction-based grasping, and a variety of object state manipulations. Each task is associated with a set of goals sampled from a continuous range of object states and their corresponding detailed task descriptions in human language form. We further provide plentiful demonstrations of each task with trajectories generated with a template-based planner for robot learning.

To provide an in-depth evaluation of language-grounded task learning, we complement prior research with an evaluation that targets the ability of agents to generalize learned language-grounded skills to unseen scenarios, including novel scenes, novel objects, and our featured novel goal states. We have meticulously curated a collection of 40 distinctive objects and 20 scenes from open-source datasets (Xiang et al., 2020; Kolve et al., 2017; Fu et al., 2021) and designed data splits for evaluating different aspects of agents' generalization ability in language-grounded task learning. Furthermore, we provide thorough experimental analyses and show that state-of-the-art language-conditioned manipulation models still suffer with regard both to grounding and generalization. Additionally, we show that state modeling is crucial for tasks in `ARNOLD` through carefully designed ablation studies.

In summary, `ARNOLD` makes the following contributions:

- A **realistic 3D interactive environment** with diverse scenes, objects, and **continuous object states**, facilitating the learning and evaluation of precise robot manipulation.
- A systematic benchmark comprising eight challenging **language-grounded robotic tasks** and evaluation splits for different aspects of **skill generalization**.
- **Extensive experiments and analyses** of state-of-the-art language-conditioned manipulation models, revealing their strengths and weaknesses in promoting future research on language-grounded task learning.

4.2 The ARNOLD Benchmark

The ARNOLD benchmark is motivated by the abilities that an intelligent manipulator agent should possess, including (1) the ability to comprehend and ground human instructions to precise world states, (2) the capacity to acquire policies for generating accurate actions and plans toward precisely defined goal states, and (3) the feasibility of transferring such abilities to the real world. Therefore, in ARNOLD we focus on language-conditioned manipulation driven by continuous goal states situated in diverse photo-realistic and physically-realistic 3D scenes.

4.2.1 Simulation Environment

Simulation Platform: ARNOLD is built on NVIDIA's Isaac Sim (Makoviychuk et al., 2021), a robotic simulation application that provides photo-realistic and physically-accurate simulations for robotics research and development. In ARNOLD, the photo-realistic rendering is powered by GPU-enabled ray tracing, and the physics simulation is based on PhysX 5.0. Figure 1.1 and Figure 4.1 provide examples of simulation and rendering.

Physical Simulation: To ensure physically-realistic simulation, we assign physics parameters to objects, including weight and friction for rigid-body objects, and cohesion, surface tension, and viscosity for fluids. These parameters are selected as in prior work (Mu et al., 2021) and are adjusted by human operator feedback. Fluids are simulated using the GPU-accelerated position-based-dynamics (PBD) method (Macklin and Müller, 2013) through NVIDIA's Omniverse platform. Depending on the rendering speed, we perform an optional surface construction process using marching cubes (Lorensen and Cline, 1987) to achieve the final fluid rendering effect.

Scene Configuration: There are 40 distinct objects and 20 diverse scenes in ARNOLD. The scenes are curated from (Fu et al., 2021), a large-scale synthetic dataset of indoor scenes. This endows ARNOLD with professionally designed layouts and high-quality 3D

models. In addition to objects provided by Isaac Sim, we collected objects from open-source datasets (Kolve et al., 2017; Xiang et al., 2020). We modified object meshes to enhance visual realism, *e.g.*, by modifying materials and adding top covers to cabinets and drawers. For more stable physics-based grasping, we performed convex decomposition to create precise collision proxies for each object. More details are found in Appendix A of (Gong et al., 2023b).

Robot: We use a 7-DoF Franka Emika Panda manipulator with a parallel gripper in `ARNOLD` for task execution. The agent has direct control over its seven joints and its gripper. We represent end-effector actions with three spatial coordinates for translation and quaternion for rotation, as it is more tractable for policy learning (Liu et al., 2022a). We utilize the built-in motion planner of Isaac Sim to transform the end-effector action back to the space of robot joints for execution. Currently, our tasks do not involve navigation, *i.e.*, the robot base remains fixed during task execution.

Visual Input: In `ARNOLD`, we use five cameras around the robot for visual input. As shown in Figure 4.1, the cameras provide various views, including front, left, robot base, and wrist. While each camera provides RGB-D input at a resolution of 128×128 by default, users can render at arbitrary resolution. Notably, unlike the deterministic rendering in prior works (Shridhar et al., 2022b; Zheng et al., 2022), the rendering in `ARNOLD` is stochastic due to the ray tracing sampling process (Shirley and Morley, 2008), which makes `ARNOLD` more realistic and challenging. In addition to the visual observation, other auxiliary observations can be accessed, *e.g.*, camera parameters, robot base pose, and part-level semantic mask. Other Omniverse sensors (*e.g.*, tactile) are excluded here since they are not required by the tasks and models. They are all available if necessary.

4.2.2 Task Design

We include eight tasks with various goal state variations in `ARNOLD`. Specifically, we focus on continuous goal states and define success ranges around them wherein robots should

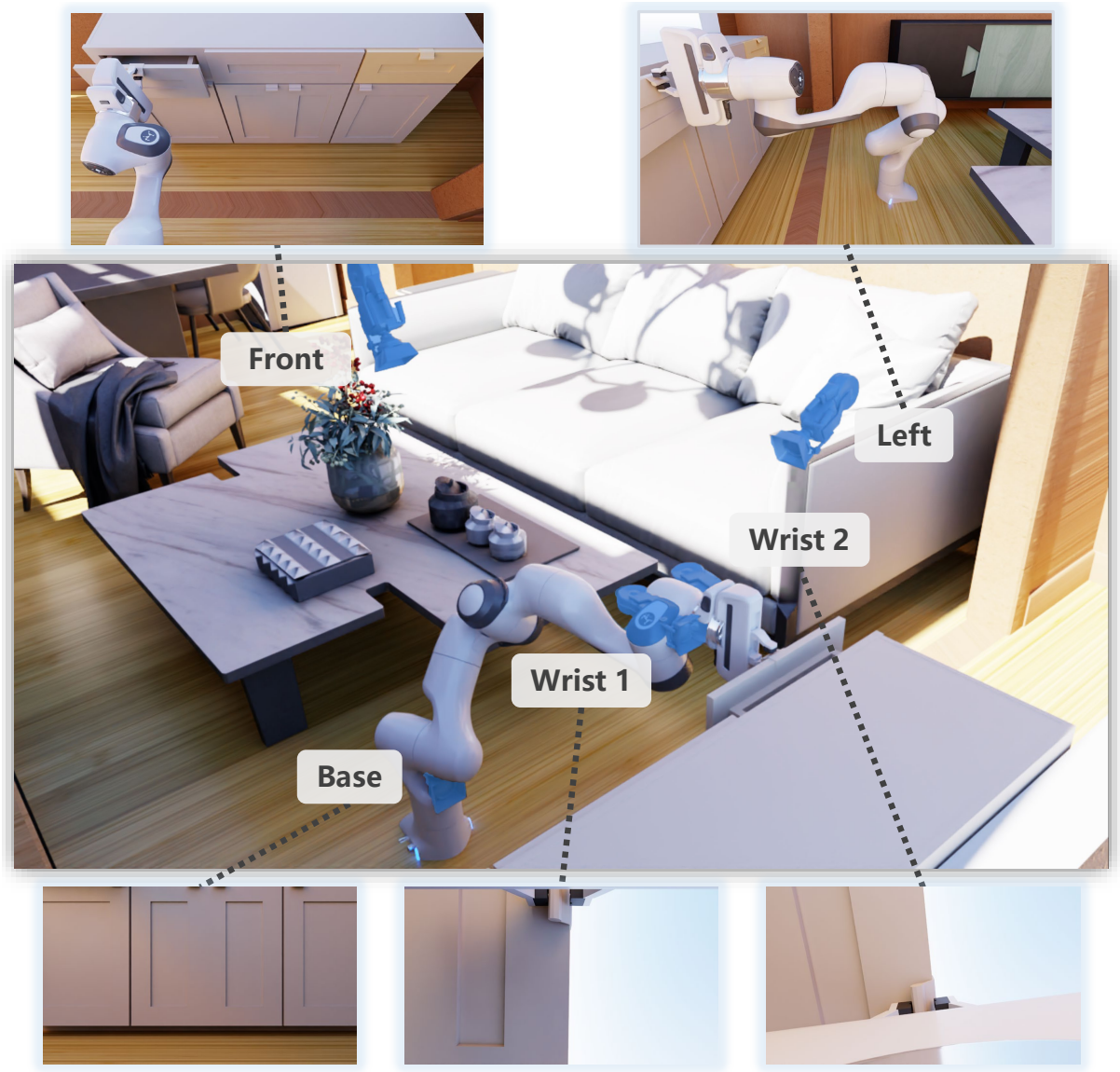


Figure 4.1: Multi-view robot observation in ARNOLD. The top row shows views from the front and left cameras and the bottom row from the base and two wrist cameras.

Task Types	Goal States	Success Ranges
PickupObject	$10, 20, 30, 40$ (cm)	± 5 cm
ReorientObject	$0, 45, 135, 180$ (°)	$\pm 20°$
OpenDrawer	$25, 50, 75, 100$ (%)	$\pm 10\%$
CloseDrawer	$0, 25, 50, 75$ (%)	$\pm 10\%$
OpenCabinet	$25, 50, 75, 100$ (%)	$\pm 10\%$
CloseCabinet	$0, 25, 50, 75$ (%)	$\pm 10\%$
PourWater	$25, 50, 75, 100$ (%)	$\pm 10\%$
TransferWater	$20, 40, 60, 80$ (%)	$\pm 10\%$

Table 4.1: Overview of the 8 tasks in `ARNOLD`. Each task features 4 goal states specified by human language, one of which is reserved for novel state evaluation. The task is deemed successful when the object state remains in the success range for two seconds. Note that TransferWater imposes the extra condition that only less than 10% of the original amount of water in the cup can be spilled.

maintain object states for 2 seconds to succeed.

Table 4.1 provides an overview and Figure 1.1 a visualization. More illustrative examples are shown in Appendix B of (Gong et al., 2023b) . Performing these tasks requires capabilities in language grounding, friction-based grasping, continuous state understanding, and robot motion planning. Additional task details follow:

- In PickupObject and ReorientObject, we instruct the robot to manipulate a bottle to achieve different goals. For the former, the initial state of the object is on the ground with goals specifying heights above the ground. For the latter, the initial state of the object is on the ground, oriented horizontally (the state value equals 90°), with goals specifying the angle between the object's orientation and the upright orientation.
- In the four tasks {Open,Close}{Drawer,Cabinet}, the goal value specifies the geometric state of the articulated joint, either in terms of distance (for prismatic joints in Drawer) or angle (for revolute joints in Cabinet). The initial state is any value smaller than the goal for Open and larger than the goal for Close.
- In PourWater and TransferWater, the manipulated object is a cup containing water, and the goal specifies the percentage of water to be poured out (Pour) or poured into another cup (Transfer). In these two tasks, the goal values are specified as percentages of water relative to the initial amount of water in the cup.

Our task pool covers a variety of manipulation skills and grounding aspects. PickupObject and ReorientObject are selected for the basic skills of moving and rotating objects and the grounding of distances and angles. These abilities are then composed and reinforced in the four tasks {Open,Close}{Drawer,Cabinet}, where the goal state is grounded on the state of the manipulated drawer or cabinet joint. Beyond rigid-body objects, fluid manipulation in the two tasks {Pour,Transfer}Water challenges the robots' ability to manipulate containers and move fluid, grounding goal state values to fluid volumes.

4.2.3 Data Collection

Demonstration Generation: We designed a motion planner for each task to generate demonstrations. We partitioned each task into sub-task stages for the planner. For each stage, we adopted the RMPflow controller (Cheng et al., 2020) to plan motions toward keypoints. Unlike other approaches to data curation in simulation environments, this keypoint-based motion planner approach affords high sampling efficiency and facilitates imitation learning. While motion planning appeared to be challenging on particular tasks, as demonstrated in (Mu et al., 2021; Gu et al., 2023), we introduced some prior design and practical techniques (details in Appendix B of (Gong et al., 2023b)) to produce satisfactory outcomes. For example, we leveraged spherical linear interpolation (Slerp) to accommodate continuous manipulation in the Cabinet and Water tasks. As a result, our motion planner can efficiently generate demonstrations.

Augmentation With Human Annotations: Despite the strength of motion planners, the diversity of produced demonstrations is highly dependent on the keypoints. To mitigate this problem, we collected about 2k human annotations of task configurations (*e.g.*, object positions), which amount to considerably more diverse and higher quality data. Moreover, we augmented the data with additional relative positions and robot shifts to broaden data variations. Eventually, we curated demonstrations by running inference with ground-truth keypoints and verifying the validity of initial configurations in each execution. In total,

	Train	*Val*	*Test*	*Object*	*Scene*	*State*	*State**	Total
PickupObject	623	134	134	275	221	294	134	1,815
ReorientObject	355	76	77	114	82	210	77	991
OpenDrawer	554	119	119	155	255	348	119	1,669
CloseDrawer	671	147	148	251	81	530	148	1,976
OpenCabinet	319	69	69	81	181	241	69	1,029
CloseCabinet	478	103	103	55	157	72	103	1,071
PourWater	312	67	67	96	87	186	67	882
TransferWater	259	56	56	51	50	119	56	647
Total	3,571	771	773	1,078	1,114	2,000	773	10,080

Table 4.2: Dataset statistics. (1) *Train*: training data. (2) *Val*: validation data for model selection. (3) *Test*: test data for i.i.d. evaluation. (4) *Object*/*Scene*/*State*: *Novel* splits for generalization evaluation. (5) *State**: the *Any State* split for generalization on arbitrary state.

we collected 10k valid demonstrations for the `ARNOLD` benchmark (as in Table 4.2), with each demonstration containing 4–6 keyframes.

Language Instructions: For each demonstration, we sampled a template-based language instruction with our language generation engine. We designed several instruction templates with blanks for each task, and each template can be lexicalized with various phrase candidates. For example, the template "*pull the* [position] [object] [percentage] *open*" may be lexicalized into "*pull the top drawer* 50% *open*". In addition to the representation by explicit numbers, we also prepared a candidate pool of equivalent phrases (*e.g.*, "*fifty percent*", "*half*", "*two quarters*") for random replacement. Note that the instruction does not specify the initial state, so the agent must recognize the current state from the observation. We present template examples in Appendix C of (Gong et al., 2023b).

4.2.4 Benchmark

Data Split: Evaluating and improving the generalization abilities of robots is a major focus of `ARNOLD`.

To this end, we randomly split the objects, scenes, and goal states into seen and unseen

subsets, respectively. We then created the *Normal* split by gathering data with seen objects, scenes, and states. The split was further shuffled and divided into *Train*/*Val*/*Test* sets proportioned at 70%/15%/15%. Notably, in addition to providing valid initialization configurations, demonstrations for evaluation splits may be used to provide intermediate ground truth for diagnosing model performance (Section 4.3.3).

Furthermore, we created the *Generalization* splits *Novel Object*/*Scene*/*State* by gathering data with one of the three components (*i.e.*, objects, scenes, and goal states) unseen; *e.g.*, the *Novel Object* split comprises data of unseen objects, and seen scenes and states.

While the *Novel State* split addresses the generalization of unseen goal states, we expect that grounding on continuous state representations should help the agent to adapt to any arbitrary state within a continuous range. Therefore, we make the *Any State* split with seen objects and scenes, setting the goal states uniformly distributed over a continuous range, *e.g.*, 0%–100%. Such a design resembles universal tasks with arbitrary goal states and facilitates the evaluation of state generalization. Table 4.2 presents the data statistics.

Metrics: A task instance is regarded as a success when the success condition is satisfied continually for 2 seconds. The success condition requires the current state to be within a tolerance threshold from the goal state; *i.e.*, the success range. The tolerances are derived according to human behaviors and are shown in Table 4.1. Note that TransferWater imposes the extra condition that only 10% or less of the water can be spilled. The execution of evaluation resembles the composition of sub-task stages in the motion planner (details in Appendix B of (Gong et al., 2023b)). To avoid accidental triggering, we check the success condition after the agent completes the final stage. For example, in the task "*pour half of the water out of the cup*", the agent succeeds if $40\% \sim 60\%$ of the water remains in the cup for 2 seconds after the agent has reoriented the cup upright. We have adopted success rate as the evaluation metric in the `ARNOLD`.

4.3 Experiments

4.3.1 Experimental Setup

Models: To evaluate the existing language-conditioned robotic manipulation models on ARNOLD, we chose two state-of-the-art models as our primary focus: 6D-CLIPort (Zheng et al., 2022) and PerAct (Shridhar et al., 2022b).

- 6D-CLIPort takes as input an RGB-D image from the top-down view and predicts end-effector poses for the current object and the target action. Each end-effector pose contains an action translation and a categorical prediction over discretized Euler angles. 6D-CLIPort comprises three branches to process the multi-modal input: Transporter-ResNet (Zeng et al., 2021) for the spatial stream, CLIP visual encoder and language encoder (Radford et al., 2021) for the semantic stream.
- PerAct takes RGB-D images as input to fuse a 3D voxelized grid. In addition, PerAct also requires the proprioception, including gripper states and the current timestep. The proprioception features are tiled on the voxel grid. Next, the hybrid voxel grid is downsampled and flattened to a sequence. Meanwhile, the language instruction is fed to a language encoder (*e.g.*, CLIP (Radford et al., 2021)) and then appended to the sequence. PerAct uses Perceiver-IO (Jaegle et al., 2022) to resample a compact latent representation from the multi-modal long sequence. After decoding, PerAct finally outputs a Q function over the original voxel grid for the prediction of action translation. Similar to 6D-CLIPort, PerAct also outputs a categorical distribution over discretized Euler angles for the prediction of action rotation. In contrast to the implementation in (Shridhar et al., 2022b), we discard the heads for predicting gripper and collision. Instead, we add an optional head for state prediction.

Moreover, we considered three model variants of PerAct in our experiments: (1) PerAct without language (PerAct w/o L) for studying the importance of language-grounding, (2) PerAct with additional supervision on state value (PerAct†) to show the urgency of state modeling for tasks in ARNOLD, and (3) PerAct trained in the multi-task setting (PerAct

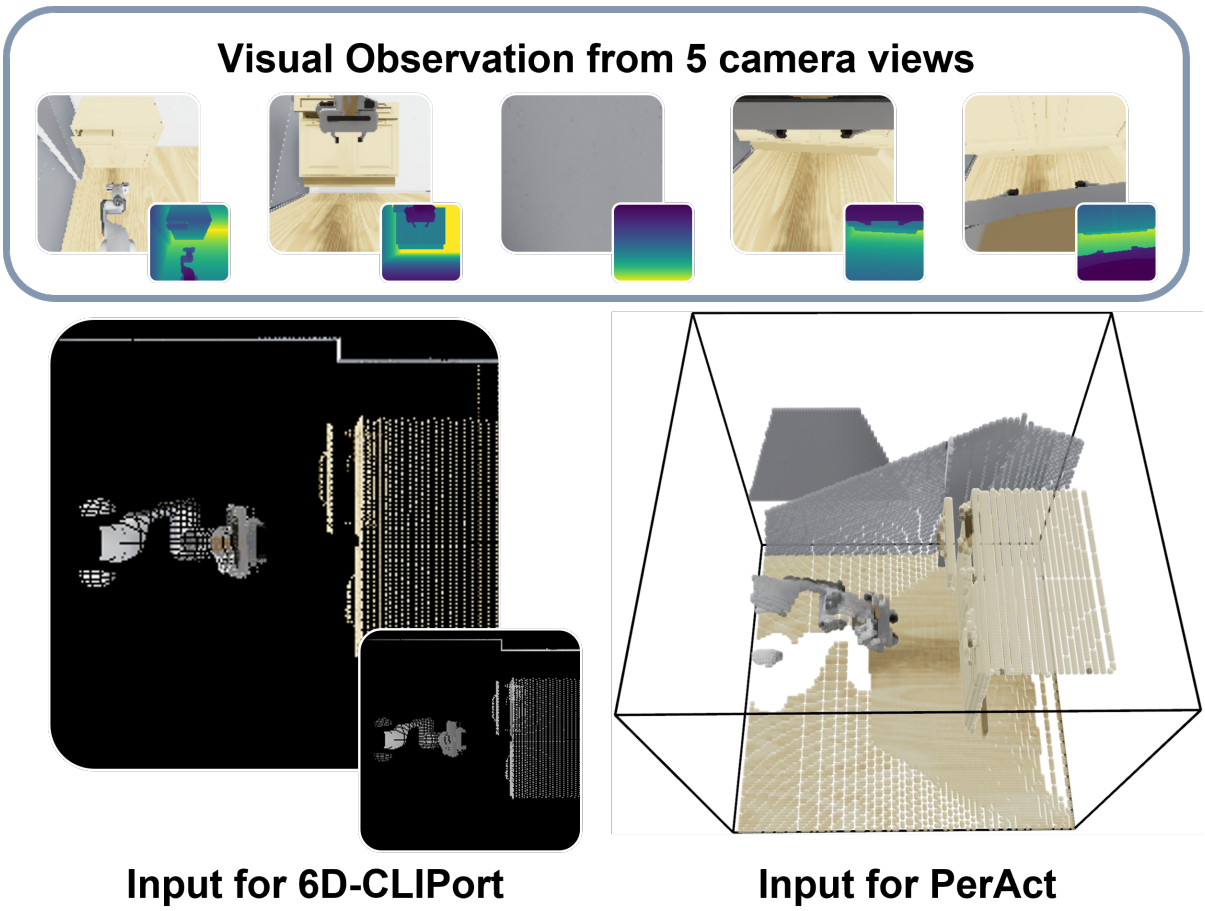


Figure 4.2: Visualization of the input representations of models. The five cameras are from the front, base, left, and two wrist views. The left camera view is occluded by the wall. Finally, these camera views fuse into an RGB-D image for 6D-CLIPort and a voxel grid for PerAct.

MT) given the great potential of multi-task learning shown in (Shridhar et al., 2022b). For PerAct†, we provide additional state supervision by adding an extra output head to regress the normalized state values from the hidden features. We also tried other manipulation models; *e.g.*, BC-Z in Appendix D of (Gong et al., 2023b).

Implementation Details: We obtain the visual representations for the models based on the five rendered RGB-D images as follows: With the camera parameters, we cast the pixels back to 3D and thus derive a point cloud for each view. With these point clouds in 3D scenes, we apply a perception bounding box to make the keypoint learning more tractable. In our setting, the cube spans 126 cm on each axis and the cube center

(p_x, p_y, p_z) is 50 cm away from the robot base along the robot's forward direction.

Next, we obtain the visual representations (visualized in Figure 4.2) as follows:

- For 6D-CLIPort, with each pixel occupying a size of 0.56 cm, we can map the 126 cm $\times$ 126 cm perceived area to a 224 $\times$ 224 top-down view image. We project the point clouds with their color and distance information onto this image. Note that the distance represents height rather than depth. If a pixel is related to multiple points (occlusion), the RGB values will be the average of these points and the distance will be the largest height.
- For PerAct, we set the size of voxel grid to be 120^3, with each voxel covering the 3D context in a 1.05 cm^3 volume. We then aggregate point clouds of all views into the voxel grid, including coordinates, RGB, positional embedding, and occupancy.

Refer to https://github.com/arnold-benchmark/arnold for detailed implementations of the models and their variants.

Learning: We trained all the models on the *Train* split and performed model selection among ten checkpoints on the *Val* split. Following common practice (Zheng et al., 2022; Shridhar et al., 2022b), we used waypoints in demonstrations to facilitate model learning. Specifically, we unify the execution of tasks in `ARNOLD` into a two-phase procedure: grasping the target object and then manipulating it toward the goal state. We followed the training settings of (Zheng et al., 2022; Shridhar et al., 2022b). Additionally, in multi-task training we sampled each training data by first uniformly sampling the task, then sampling a demonstration of the corresponding task. We set the number of training iterations to 100k for the single-task and 200k for the multi-task setting. We performed all the training on a single NVIDIA A100 GPU with batch size 8.

Evaluation Execution: The evaluation executor of each task resembles the pipeline of a motion planner yet has a slight difference (see details in Appendix B of (Gong et al., 2023b)). The model's predictions are converted back to keypoints of sub-task stages

	P.Object		R.Object		O.Drawer		C.Drawer		O.Cabinet		C.Cabinet		P.Water		T.Water		Average	
6D-CLIPort	6.72	25.37	0.00	0.00	0.00	0.00	0.00	2.70	0.00	0.00	0.00	5.83	0.00	0.00	0.00	7.14	0.84	5.13
Object	8.36	28.36	0.00	0.00	0.00	0.00	0.00	0.40	0.00	1.23	0.00	1.82	0.00	0.00	0.00	3.92	1.05	4.47
Scene	10.41	24.43	0.00	0.00	0.00	1.57	0.00	0.00	0.00	0.55	1.27	1.27	0.00	0.00	0.00	12.00	1.46	4.98
State	0.00	0.00	0.00	0.00	0.00	0.57	0.75	1.13	0.00	0.83	0.00	2.78	0.00	0.00	0.00	16.81	0.09	2.77
Any State	10.45	29.10	1.30	2.60	0.84	0.00	0.68	1.35	0.00	5.80	0.00	2.91	0.00	1.49	0.00	7.14	1.66	6.30
PerAct (w/o L)	25.37	33.58	14.29	7.79	17.65	36.13	47.30	52.03	8.70	34.78	7.77	10.68	14.93	11.94	5.36	14.29	17.67	25.15
Object	29.09	26.55	8.77	3.51	3.87	20.00	24.70	**32.67**	0.00	0.00	**1.82**	**7.27**	16.67	29.17	9.80	**19.61**	11.84	17.35
Scene	26.70	24.89	14.63	14.63	19.61	33.33	48.15	54.32	**1.10**	3.87	1.91	1.27	13.79	20.69	6.00	16.00	16.49	21.13
State	0.34	0.00	0.00	0.95	9.20	8.05	1.70	2.08	0.00	2.07	1.39	1.39	1.08	2.69	5.04	**8.40**	2.34	3.20
Any State	20.15	19.40	12.99	12.99	13.45	30.25	20.95	24.32	**5.80**	**26.09**	**14.56**	**15.53**	14.93	16.42	1.79	7.14	13.08	19.02
PerAct	94.03	**97.76**	19.48	24.68	31.09	44.54	**60.81**	66.22	**24.64**	42.03	22.33	45.63	**55.22**	74.63	**32.14**	46.43	42.47	55.24
Object	86.55	**92.73**	**11.40**	**35.09**	**6.45**	21.29	**26.29**	27.89	0.00	0.00	**1.82**	5.45	**36.46**	**42.71**	**13.73**	13.73	**22.84**	**29.86**
Scene	**72.85**	**84.62**	**17.07**	31.71	20.78	31.37	**66.67**	64.20	0.00	**4.97**	5.10	19.11	**33.33**	51.72	22.00	**36.00**	**29.73**	40.46
State	2.38	0.68	0.00	0.95	**10.92**	12.64	**13.77**	17.74	0.00	5.81	1.39	1.39	1.61	1.08	**5.88**	1.68	4.49	5.25
Any State	47.01	**50.75**	7.79	**19.48**	**21.85**	30.25	18.92	25.00	**5.80**	21.74	3.88	**15.53**	14.93	25.37	10.71	14.29	16.36	25.30
PerAct†	**94.78**	95.52	**24.68**	**28.57**	**36.13**	**52.94**	60.14	**68.24**	23.19	**49.28**	**30.10**	**48.54**	49.25	**85.07**	28.57	**53.57**	**43.36**	**60.22**
Object	**87.27**	91.27	10.53	32.46	1.94	**22.58**	18.73	25.10	0.00	**4.94**	0.00	5.45	34.38	33.33	9.80	11.76	20.33	28.36
Scene	69.68	84.16	13.41	37.80	**25.49**	**38.43**	60.49	**67.90**	0.55	**4.97**	6.37	**19.75**	29.89	**63.22**	**26.00**	24.00	28.99	**42.53**
State	0.68	2.38	0.48	0.00	10.06	12.93	13.58	**18.11**	0.00	**6.22**	0.00	8.33	2.15	1.61	**5.88**	2.52	4.10	6.51
Any State	**48.51**	47.76	**14.29**	14.29	21.01	**33.61**	**23.65**	**28.38**	4.35	24.64	6.80	13.59	**26.87**	**31.34**	**14.29**	**19.64**	**19.97**	**26.66**
PerAct (MT)	88.81	88.81	3.90	22.08	26.05	43.70	33.78	52.03	11.59	33.33	20.39	37.86	34.33	58.21	14.29	23.21	29.14	44.90
Object	77.09	77.45	7.02	20.18	1.29	15.48	13.55	25.50	0.00	0.00	**1.82**	**7.27**	13.54	30.21	1.96	11.76	14.53	23.48
Scene	68.78	77.83	10.98	**41.46**	13.33	25.88	29.63	55.56	0.00	**4.97**	5.73	9.55	19.54	43.68	4.00	16.00	19.00	34.37
State	**9.18**	**12.59**	0.00	1.90	8.05	12.07	9.43	13.40	0.00	2.07	1.39	0.00	2.69	6.45	**5.88**	5.88	**4.58**	6.80
Any State	36.57	45.52	5.19	10.39	15.13	19.33	16.22	21.62	1.45	8.70	0.97	3.88	8.96	14.93	3.57	17.86	11.01	17.78
PerAct (MT)†	90.30	92.54	14.29	20.78	25.21	47.90	33.78	56.76	20.29	39.13	19.42	37.86	26.87	64.18	17.86	30.36	31.00	48.69
Object	81.09	85.45	7.89	24.56	3.23	**22.58**	14.74	28.69	0.00	**4.94**	**1.82**	5.45	9.38	20.83	3.92	**19.61**	15.26	26.51
Scene	67.87	79.64	7.32	29.27	12.94	25.49	39.51	65.43	**1.10**	4.42	**7.01**	14.01	10.34	43.68	8.00	22.00	19.26	35.49
State	2.04	3.06	**0.95**	**2.38**	9.20	**18.68**	6.98	11.13	0.00	3.73	**2.78**	**11.11**	**6.45**	**9.14**	1.68	4.20	3.76	**7.93**
Any State	46.27	47.01	12.99	12.99	12.61	23.53	14.86	26.35	4.35	5.80	4.85	8.74	16.42	25.37	3.57	5.36	14.49	19.39

Table 4.3: Evaluation results of the models on various tasks and splits, measured by success rate and shown in percentages. The gray figures indicate performances with the first-phase ground truth. For each model, the first row shows the performance on the *Test* set, and the following three rows show those on the *Novel* splits of *Object*, *Scene*, and *State*. The last row indicates the performances on the *Any State* split. Tasks are abbreviated for more space. Average performances on eight tasks are appended to each row. **w/o L**: without language instruction. †: model variants with state modeling. **MT**: multi-task models.

for evaluation execution. We make our evaluation strict and appropriate by avoiding shortcuts on the object state during robot motion. For example, the task instructing "*pull the cabinet* 50% *open*" will not be considered a success even if the cabinet is held around 50% open for 2 seconds so long as the motion planner has not executed its final action. To eliminate the influence of a first-phase failure (*i.e.*, failing to grasp the target object or object part), we conduct additional evaluations that provide first-phase ground truth.

4.3.2 Experimental Results

We report experimental results in Table 4.3 (black) and present our findings and analyses below.

Across Models: Comparing the baseline models 6D-CLIPort, PerAct, and PerAct (MT), we found that 6D-CLIPort fails on most of the tasks. We conjecture that this stems from (1) the information lost in the input representation when compressing complex 3D scenes into a single image and (2) the difficulty in regressing target height values for action translation. By contrast, the voxelized representations in PerAct provide rich 3D contexts that benefit model learning. With such differences, PerAct outperforms 6D-CLIPort significantly. Meanwhile, we observed performance drops for PerAct (MT) compared to PerAct on all tasks. This indicates that it is still difficult to leverage more diverse multi-task data for efficiently learning better task policies in `ARNOLD`, especially given its challenges in both grounding and manipulation.

Across Tasks: The most challenging tasks in `ARNOLD` are ReorientObject and {Open,Close}Cabinet. ReorientObject is difficult because it involves estimation of the bottle orientations, and models can often be confused by visually similar states. For example, the action for the goal state of 45° will lead to a goal state of 135° if the bottle orientation is reversed. Manipulating a cabinet proved to be challenging even with privileged information to specify goals (Mu et al., 2021; Gu et al., 2023) as it requires accurate prediction of both interacting position, rotation, and precise continuous motion control. Replacing privileged information with instructions will only make it harder. Furthermore, we observed superior model performance in PourWater compared to TransferWater. This is because transferring water requires position alignment between cups to avoid spillage.

On Generalization Splits: In general, we observed performance drops for most models when transferring to *Novel* generalization splits, especially on the *Novel State* split. This reveals that, without proper modeling of continuous states, generalizing the grounding of seen goal states to unseen ones remains challenging. Meanwhile, the performance drop on the *Novel Object* and *Novel Scene* splits varies according to the tasks. For tasks where the objects occupy substantial space (*e.g.*, drawer), the impact of unseen objects is more

significant than unseen scenes.

For the *Any State* split, the models' performances were inferior compared with the *Novel Object/Scene* split and superior to those on the *Novel State* split. As goal states are uniformly sampled from a continuous spectrum, the success ranges of seen goal states are likely to cover a large portion of the spectrum, making the *Any State* generalization interpolations of learned knowledge and skills. This suggests an interesting research question that can be investigated with `ARNOLD`: How can the task learning model better generalize by interpolating within ranges and extrapolating to out-of-range goal states?

Remarks: The key findings from our experiments with `ARNOLD` are as follows:

- Current models still struggle with tasks that require complex manipulation skills (*e.g.*, manipulating cabinets). *This heightens the demand for better policy learning models to tackle challenging manipulation tasks.*
- The low success rate of models on all generalization splits motivates the necessity for (1) *increasingly fine-grained representations for perceptual inputs*, (2) *finer modeling of continuous object states*, and (3) *better alignment between language and robot actions.*
- The *Any state* experiments suggest that *state generalization could potentially be achieved through the interpolation of acquired knowledge and skills*. This promotes approaches with deeper insights into systematic generalization for robot skill adaptation.

4.3.3 Ablation Studies

Influence of Language: PerAct (w/o L), trained with a single-task scheme, exhibits a considerable performance gap behind PerAct. This indicates the importance of the goal-state information in `ARNOLD`. Meanwhile, we observe a relatively small gap for ReorientObject. We believe this is due to (1) the bottleneck of this task lying in the ambiguity of visual perception, as discussed in Section 4.3.2 and (2) the difficulty of grounding angles from current visual observations. On the other hand, the significant performance gap on PickupObject shows the effectiveness of language grounding in

visually more identifiable concepts such as translation distance.

Importance of State Modeling: The PerAct variants with state supervision (†) were expected to realize a performance gain from explicit state prediction supervision. However, we observed marginal improvements on the *Test* split and limited enhancements on generalization splits for this method. This indicates that such end-to-end state supervision is insufficient for state modeling in ARNOLD and calls for better approaches to representing and modeling the continuous object states in robotic task learning.

With Intermediate Oracle: To better demonstrate how well models understand goal states, we provided models with first-phase ground truth (*i.e.*, grasping positions) and report their performance in Table 4.3 (gray). As expected, we observed significant performance gains with such ground truth. Moreover, directly comparing the scores with the first-phase oracle, we can also observe larger gaps between PerAct and PerAct†, as well as PerAct (MT) and PerAct (MT)†. These results demonstrate that explicit state modeling is indeed beneficial for goal state understanding.

Choice of Language Encoder: We investigated model ablation over the language encoder by switching the default language encoder CLIP (Radford et al., 2021) in PerAct to T5-base (Raffel et al., 2020). Due to space constraints, we report model performance only on OpenDrawer. As shown in Figure 4.3, PerAct-T5 outperforms PerAct-CLIP on all the benchmarking splits. This may be due to the inefficacy of the global language representation learned in CLIP in representing fine-grained goal states for precise control. By contrast, T5 is a general-purpose language processing model that offers the ability to maintain detailed information through language modeling and generation. This shows that finer language embeddings may benefit the language grounding of robots to some extent.

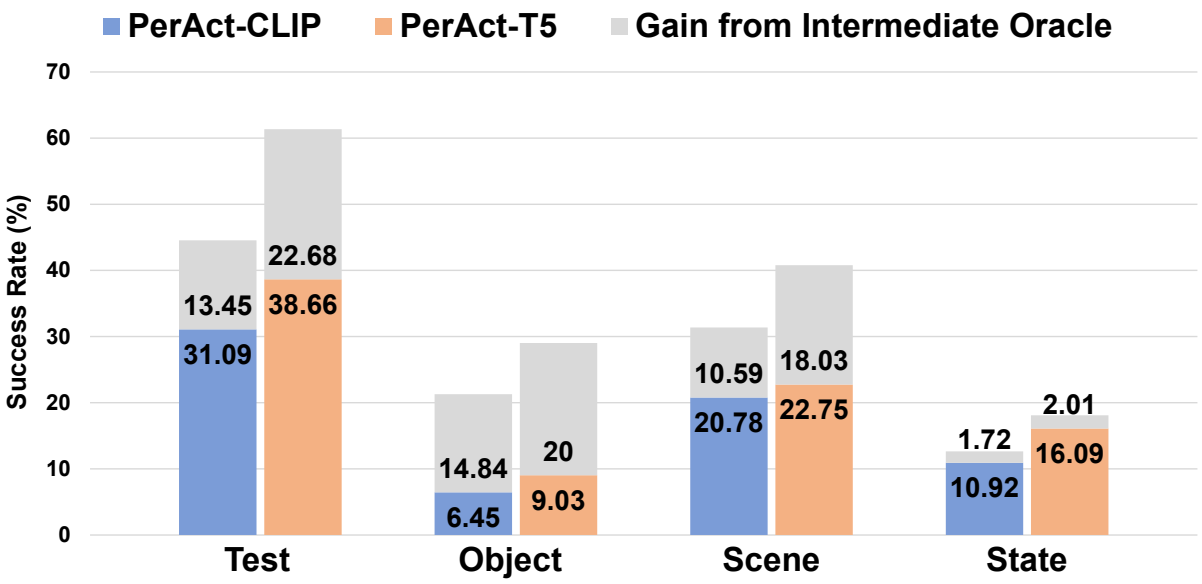


Figure 4.3: Model ablation results with different language encoders

4.3.4 Sim2Real Transfer Experiment

As a realistic simulation environment, one key question to address is: *To what extent can the agents trained in* `ARNOLD` *generalize to real-world scenarios*? To this end, we set up a real-world environment for testing the Sim2Real transfer capabilities of agents. Specifically, we used the Franka robot arm to manipulate previously unseen real-world objects with partial point cloud observations captured through a single RGB-D camera from the left view (Figure 4.1). We experimented with PerAct, which was trained in `ARNOLD` to open/close 2 different drawers and pick up 5 different objects. We mitigated the Sim2Real gap as follows:

1. Perception: We used high-fidelity rendering. Due to the imperfect depth sensor in the real world, we sprayed contrast aiding paint onto metallic and transparent areas. For diffuse objects with acceptable depth quality, further domain adaptation would be beneficial.
2. Control: Instead of using the qpos control API, we used an inverse kinematics (IK) controller to perform real-robot actions, which avoids error accumulation.

Additional experimental details can be found in Appendix D of (Gong et al., 2023b).

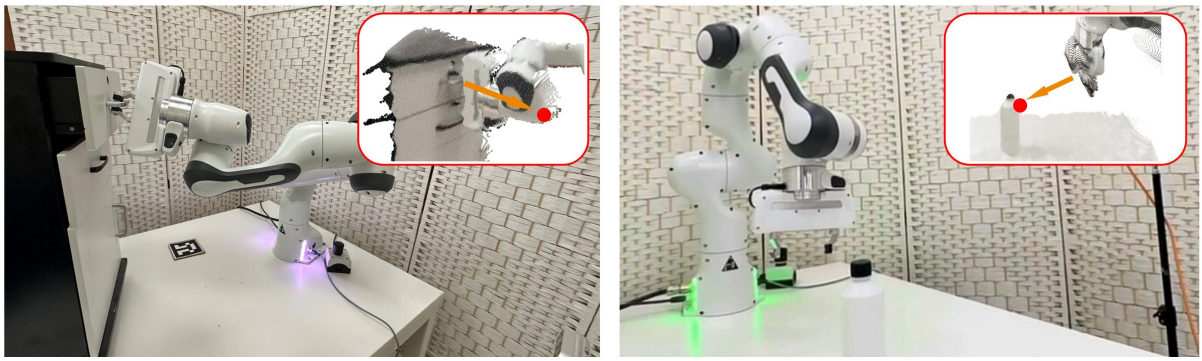


Figure 4.4: Real-world experiments with inference results shown on the upper right. The red dots indicate the predicted positions of the next action for the end-effector.

Throughout our experiments, we observed that models trained in ARNOLD show preliminary Sim2Real transfer capabilities; *i.e.*, reasonable predictions for both picking up objects and manipulating drawers, as shown in Figure 4.4. However, the actual robot manipulation continues to struggle because of the Sim2Real gap. For example, when opening a non-plastic drawer in the real world, the robot encounters high friction and is therefore susceptible to prediction errors that lead to inexecutable actions (*e.g.*, exceeding the critical friction angle). With more fine-grained object assets, we believe that the flexible design of the ARNOLD simulator can gradually close this Sim2Real gap by providing more realistic simulations.

4.4 Summary

We have presented ARNOLD, a benchmark for language-grounded task learning in realistic 3D interactive environments with diverse scenes, objects, and continuous object states. We devised a systematic benchmark comprising eight challenging language-grounded robot tasks and evaluation splits for robot skill generalization in novel scene, object, and goal-state scenarios. We conducted extensive experiments and analyses to pinpoint the limitations of the current models and identified promising research directions for grounded task learning.

Part II

Multi-Agent Collaboration

CHAPTER 5

Joint Mind Modeling for Explanation Generation in Complex Human-Robot Collaborative Tasks

5.1 Introduction

In recent years, there has been a great amount of success on building powerful artificial intelligence (AI) systems to solve complex tasks (Levine et al., 2016; Bansal et al., 2017). As highly autonomous robots are being developed, there is a growing need to make them quickly understood to avoid consequences caused by misunderstanding (Gunning, 2017). However, existing robot systems are often not human compatible — i) they do not understand humans' minds and ii) they are just black boxes to humans too. Such limits prevent the AI systems from working with humans effectively.

Inspired by studies on the Theory-of-Mind (Premack and Woodruff, 1978; Dennett, 1989), we believe that a crucial step towards building human compatible systems, particularly for human-robot collaborations, is to understand human activities and their underlying mental state. As a motivating example, consider a robot chef helping a human make salads in the kitchen shown in Figure 5.1. Even when the robot understands how to perform the task on its own, it would be challenging to finish the task efficiently without having a shared mental model with its human partner. For making the salad, the robot believes the plate should be picked up by the user while the human agent believes the other way. If the robot can identify such discrepancies between different agents' mental states, it can generate explanations to mitigate the differences and encourage the correction of sub-optimal human behavior.

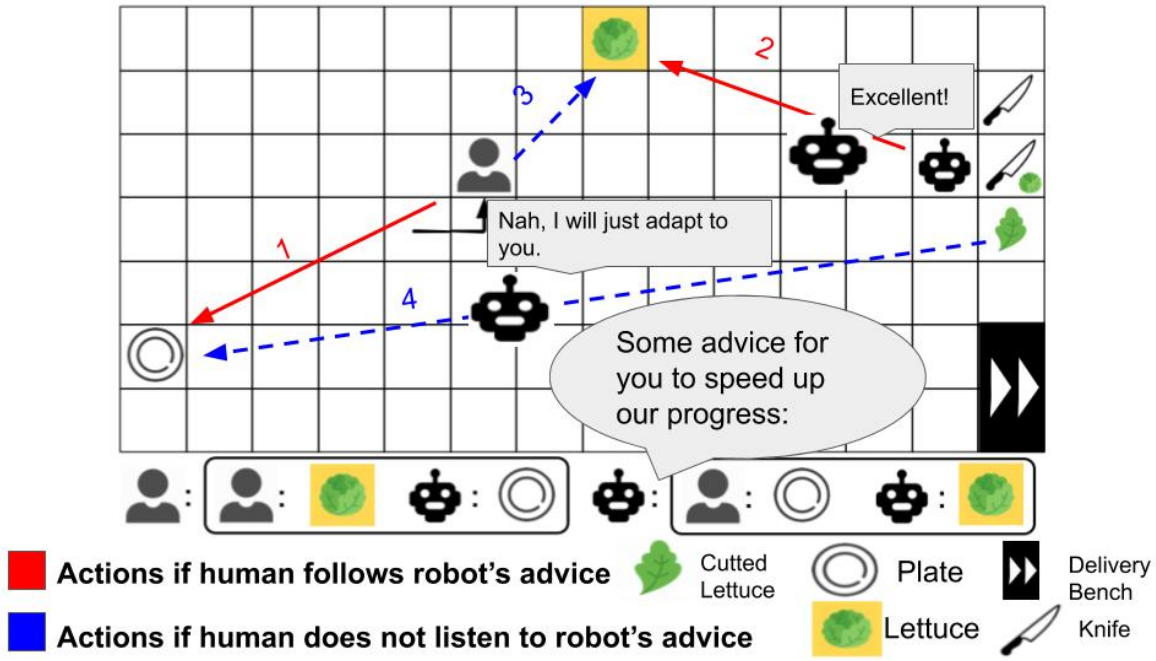


Figure 5.1: The task *making salad* requires team members to take three lettuce from the basket and cut each one with a knife, before it can be put into the plate and served. After the first lettuce has been cut, the robot is cutting the second one. The robot can identify human's sub-optimal behavior (taking new lettuce from the basket) before generating explanations to the human.

To this end, we propose a framework that improves human-robot teaming performance through explanations. With a graph-based representation, the robot can maintain the mental states of both team members during a highly-structured collaborative task. The robot can then generate explanations when difference between mental states is detected, which implies sub-optimal user behaviors. In summary, the main contribution of this chapter is three-fold:

- We design a real-time collaborative cooking game as an online user study system and develop an evaluation protocol, which can be accessed from our website.
- We propose to understand complex human activities using an action parsing algorithm based on an And-Or graph task representation, which allows the robot to infer human mental states in complex environments.
- Based on the inferred human mental state, we propose an explanation generation

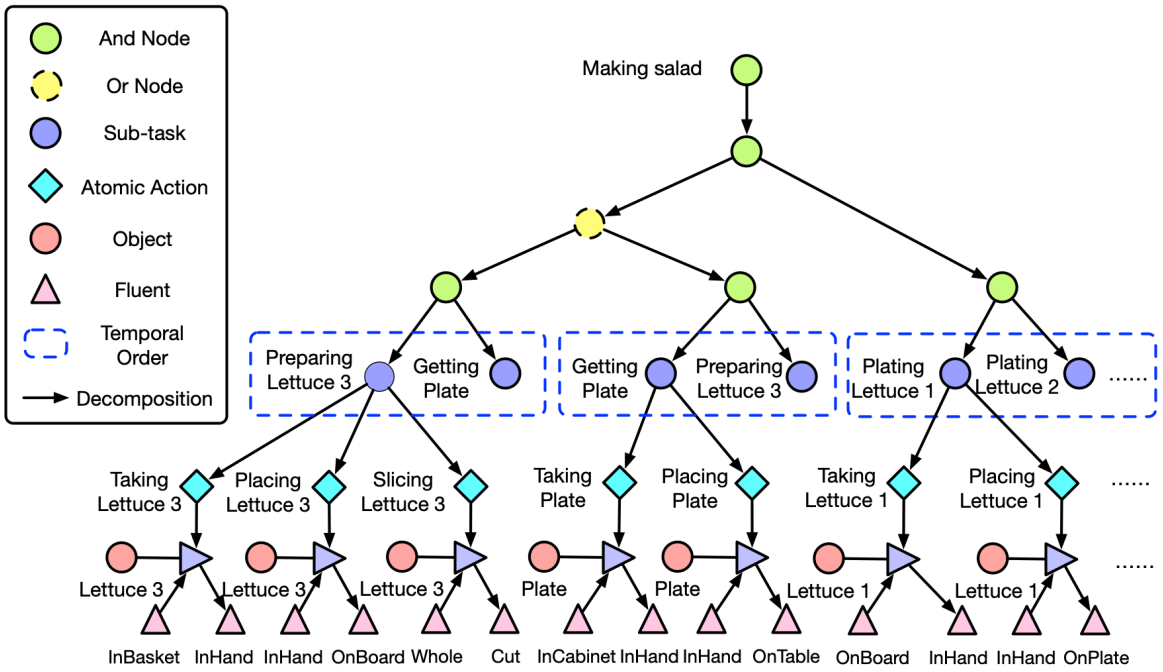


Figure 5.2: The hierarchical mind model for the collaboration task, "making salad", represented by an AoG. The And node represents temporal relations between sub-tasks. The Or node represents two possible ways for the team to finish the tasks. Each terminal node (diamond) denotes an atomic action that would cause certain fluent changes (triangles) for objects.

framework. Experiments on a real-time cooking task show that our approach successfully improves user perception of the robot and leads to better human-robot collaborations.

5.2 Single Agent Mind Model

And-Or graphs (AoGs) have been widely used for robot task planning (Xiong et al., 2016; Shu et al., 2017; Liu et al., 2018) and human activity modeling (Tu et al., 2013; Shu et al., 2015). As a hierarchical representation, a spatial-temporal-causal And-Or graph (STC-AoG) encodes a joint task plan and corresponding spatial, temporal, and causal relations an agent could have about the task (Xiong et al., 2016). In this work, we propose to use a STC-AoG as a unified representation of a robot's knowledge and plan regarding the task as well as the inferred human's knowledge and plan. An example of a single-agent plan for *making salad* is in Figure 5.2.

5.2.1 STC-AoG as a Hierarchical Mind Model

In general, an And-Or Graph consists of nodes and edges. The set of nodes includes Or node, And node, and Terminal node. Each **Or node** specifies the Or relation: only one of its children nodes would be performed at a given time. An **And node** represents the And relation and is composed of several children nodes. Each **Terminal node** represents a set of entities that cannot be further decomposed. The edge represents the top-down sampling process from a parent node to its children nodes. The root node of the And-Or tree is always an And node connected to a set of And/Or nodes. Each And-node represents a sub-task which can be further decomposed into a series of sub-tasks or atomic actions.

In this chapter, the graph $G =< A, F, T, V, R, P >$ is formally defined as the following:

- A is a set of terminal nodes. Each node corresponds to an atomic action $a \in A$.
- F is a set of object states essential to the task, including possible pre-conditions and post-effects of atomic actions.
- $T : F \times A \rightarrow F$ is a set of transition rules that represent state changes caused by atomic actions.
- V is a set of non-terminal nodes, which can be further decomposed into two sets: the And nodes S and the Or nodes O. Each sub-task corresponds to an And node s, which encodes a temporal relationship between its children. An Or node o forms a production rule with an associated probability; *i.e.*, you may choose one of its children each weighted with a certain probability.
- R is the set of production rules.
- P is the set of probabilities on production rules.

Causal Relation: Causal knowledge represents the pre-conditions and the post-effects of atomic actions. We define it as a fluent change caused by an action. Fluent $f \in F$ can be viewed as some essential properties in a state x that can change over time; *e.g.*,

the temperature in a room and the status of a heater. For each atomic action, there are pre-conditions characterized by certain fluents of the states; *e.g.*, an agent cannot successfully turn on the heater unless it is plugged in. As the effect of an action, certain fluents would be changed, and the state x would evolve to x'. For example, if someone turns on a heater, the temperature of the room will be higher (and the heater would be on). It is formulated as one of the transition rules T.

Temporal Relation: Temporal knowledge encodes the schedule for an agent to finish each sub-task. It also contains the temporal relations between atomic actions in a low level sub-task. The sub-task preparing salad, for example, consists of taking salad, placing it onto the cutting board, and using the knife.

Spatial Relation: Spatial knowledge represents the physical configuration of the environment that is necessary to finish the task. In our case, to make the salad, an agent needs to know the locations of ingredients (*e.g.*, lettuce), tool benches (*e.g.*, basket, cutting board), delivery benches, *etc.*.

5.2.2 Parse Graphs as Mental State Representations

During the collaboration, an agent can use parse graphs to represent the mental states of itself or the other agent. A parse graph is an instance of an And-Or Graph, each of its Or nodes selects one child node. Figure 5.3 shows two parse graphs represent the robot and human's plan for the situation is shown in Figure 5.1. In our case, the parse graph $pg_t =< s_t^h, s_t^r, a_t^h, a_t^r, f_t^h, f_t^r >$ is one possible plan for both agents to finish the task. Particularly, the root node leads to a selection of individual sub-tasks (s_t^h, s_t^r) as sub-goals assigned to human and robot agent. To achieve these sub-goals, agents perform atomic action (a_t^h, a_t^r) based on their belief of current fluent (f_t^h, f_t^r).

5.2.3 Joint Task Planning by Parsing STC-AoG

To construct the mental state representation for the robot, we design an algorithm based on STC-AoG parsing to select the optimal task plan for the team.

Given a set of sub-tasks S necessary to complete the joint task, the objective is to minimize the total task completion time by assigning a sub-task to either a robot or human agent, without violating any latent constraint:

$$\begin{aligned} &\min_{x_s^v, \tau_s} \max_{v \in \{r,h\}} \sum_{s \in S} x_s^v \delta_s^v \\ &\text{s.t. } x_s^v \in X_{\text{feasible}}, \tau_s \in \Gamma_{\text{feasible}}, \end{aligned} \tag{5.1}$$

where x_s^v is a binary variable indicating whether to assign sub-task s to agent v, and τ_s is a continuous variable representing the finishing time for the sub-task s. Constant δ_s^v represents the amount of time for agent v to finish the sub-task s. X_{feasible} and Γ_{feasible} represent the set of valid assignments that satisfies latent causal constraints, *e.g.*, an agent cannot hold two objects at the same time; a sub-task can be performed only if pre-conditions are met; after all assigned sub-tasks have been completed, the final state should satisfy the goal requirement.

We search for the optimal task plan via a dynamic programming algorithm. Starting from the initial state f_b, we make valid sub-tasks assignments and simulate new intermediate state f_e based on the state transition function T. By updating the current optimal consumed time and the corresponding sub-task assignment vectors for every intermediate state, our algorithm will finally reach the optimal plan for the entire task. During the updating process, we also record the sub-task assignment vectors for previous states, in order to generate the whole optimal assignment $\{x_s^v\}_{s=1,...,|S|}$ and completion time $\tau_1, ..., \tau_{|S|}$ for each sub-task. After the task plan is computed, the robot's mental model is represented by a parse graph, as shown in the left part of Figure 5.3: each sub-task in the task plan indicates a sub-goal that an agent needs to achieve at the time being. Sub-tasks are further connected with a sequence of corresponding atomic actions, which

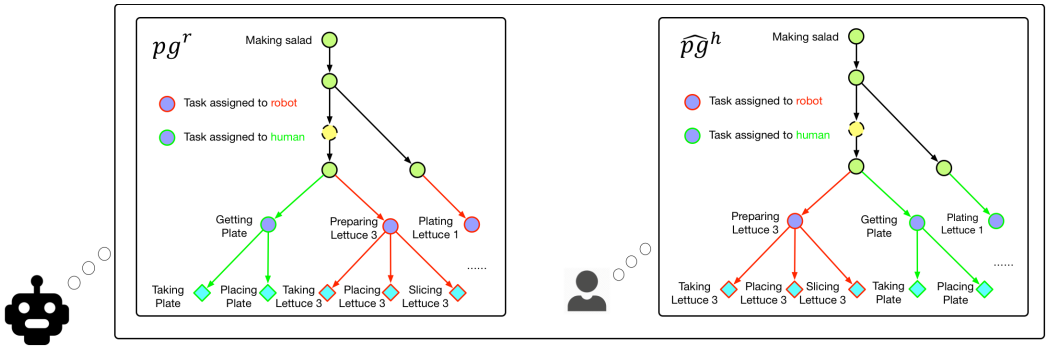


Figure 5.3: Robot mental state pg^r and inferred human mental state $\hat{pg}^h$ represented as parse graphs.

have certain pre-conditions and post-effects.

5.3 Joint Mind Modeling for Human-Robot Collaborations

Our goal is to enable efficient human-aware collaboration for a human-robot team. Specifically, robots need to understand human agents based on their actions and decide whether the team is moving in the right direction. We propose to model the robot mental state pg^r and the human mental state pg^h.

5.3.1 Mind Models for Human and Robot

We treat the robot's mind as the oracle; *i.e.*, it contains all necessary spatial, temporal, and causal information the team needs to finish the task. For example, at any given time t, the robot has a certain expectation of (i) current low level sub-goals (s_t^h, s_t^r) both agents should be pursuing; (ii) the actions (a_t^h, a_t^r) agents should perform; (iii) whether current object fluents satisfy pre-conditions of such actions, and what would be the post-effects.

It is also necessary to model the user's mind, which acts as a strong inductive bias in predicting user activities. As the user's mental state pg_t^h is not directly available to the robot, we propose to infer it from user behavior and the history of communication.

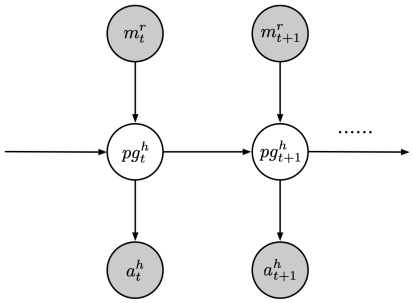


Figure 5.4: Human mental model update process. We use it to infer user mental state pg^h, which is hidden to the robot. Here we assume human actions a_t^h and robot message m_t^r are conditional independent given human mental state pg_t^h at time t.

5.3.2 Human Mental State Inference

Based on the observed user behavior, we infer the most likely human mental state $\hat{pg}^h$, including the belief, goal and action plans. On a high level, this inference process uses observed user actions and communication history to infer human mental state. Specifically, given the And-Or graph G and human-robot interaction data $D_T = \{d_t\}_{t=1,\dots,T}$, we infer the user mind $\hat{pg}^h$ iteratively:

$$\hat{pg}^h = \arg\max_{pg^h} p(pg^h|D_T, G), \tag{5.2}$$

$$p(pg^h|D_T, G) \propto p(pg^h|G, D_{T-1})p(d_T|pg^h, G). \tag{5.3}$$

Here the first term models the prior on the user mind given previous data D_{T-1} and AoG structure G. The second term models the likelihood for new data d_T.

To model the likelihood function $p(d_T|pg^h, G)$, we take a sampling-based approach. For each interaction data d, we consider user atomic action a_{obs}^h and communication between the two agents m. The idea is to model how likely the user performs action a_{obs}^h when receiving message from the robot m^r, with current mental state pg^h, as shown in

Figure 5.4. Assuming a^h_{obs} and m^r are conditional independent given pg^h we have:

$$p(d|pg^h, G) = p(a^h_{obs}|pg^h, G)p(m^r|pg^h, G), \tag{5.4}$$

$$p(a^h_{obs}|pg^h, G) = \sum_{a^h_{samp}} p(a^h_{samp}|pg^h)p(a^h_{obs}|a^h_{samp}), \tag{5.5}$$

where $p(a^h_{samp}|pg^h)$ denotes the probability of sampled human action a^h_{samp} given current estimation of human mental state pg^h. $p(a^h_{obs}|a^h_{samp})$ measures the similarity between observed human trajectory a^h_{obs} and sampled trajectory a^h_{samp}.

In practice, we use rapid-exploring random tree (RRT*) for trajectory sampling and dynamic time warping (DTW) based approach to compare trajectories. DTW outputs a difference score $diff$. We use it in the energy function for the Boltzmann distribution. Then we update the human mental state in every time-step through the following equation:

$$P(\hat{pg}^h_{t+1}|D_T, G) = \frac{1}{Z} e^{-\frac{diff}{T}} \lambda^n P(\hat{pg}^h_t|D_{T-1}, G), \tag{5.6}$$

where T is a constant temperature term, Z is a normalization constant, and λ (> 1) is a constant that controls the importance of an explanation. It models how much information the user can retain for an explanation. n is the number of times an explanation about $\hat{pg}^h$ is generated for the user in this task. Therefore, λ^n implicitly encodes the communication history m. Right now, we only consider communications from robot to human m^r. Communication from human to robot m^h can be considered in the future by adding corresponding energy terms. For now, some parameters (T and λ) are set heuristically. These parameters can be learned from annotated user data (Carreira-Perpinan and Hinton, 2005).

5.3.3 Robot Mental State Update

Based on the observations in the environment, the robot can update its joint task plan. It is a two-step process. First, the robot collects all relevant information about the task and calls a DP planner described in Section 5.2.3 to obtain an optimal sequence of sub-tasks. Then the robot updates its mental state through re-organizing AoG (Delete finished nodes. Re-order unfinished nodes. If necessary, add back nodes deleted previously). Second, the robot uses causal knowledge (pre-conditions and post-effects of each atomic action) in the AoG terminal nodes to determine the next atomic action. If pre-conditions for the next atomic action are satisfied, the robot will execute it. Otherwise, the robot will be idle, waiting for the user to complete the other part of the job.

5.4 Explanation-Based Task Coaching

In this section, we propose a framework for explanation generation to enable efficient human-robot collaboration.

5.4.1 Explanation Framework

As shown in Algorithm 1, the framework includes an iterative process of online planning and explanation generation:

1. At a given time, the robot updates its mental state to represent the expected current goals of both agents and corresponding atomic actions;

2. The mental state of the human agent can be inferred, which would be further compared to the robot's mental state. Based on the result, the robot would decide whether explanations are necessary;

3. On the occasions where users perform an action other than that indicated in the explanation, the robot would update its task plan and mental model to reflect the best joint policy and expected mental models in the new state.

Algorithm 1: Planning and explanation generation

```
while Task not finished do
    if Replan needed then
        Collect state information from the game;
        Collect predicted human intentions from the last time step ;
        Call DP planner ;
        Obtain a new sequence of sub-tasks from planner and re-organize AoG
         based on it;
        Parse AoG through checking pre-conditions and post-effects against the
         current environment state information ;
        Find out the next atomic action to execute based on parsing result ;
    Predict human intentions by (5.6) ;
    Measure the difference between predicted intention and expected human
     actions;
    Generate an explanation if difference is significant;
```

Take the task *making salad* for example. At the beginning of the game, an optimal plan requires the user to first take the plate. A sub-optimal plan could be the user first taking the lettuce. If the user insists on taking the lettuce first regardless of whether explanations are given, the robot will update the task plan and expect the user to gather the plate afterwards.

5.4.2 Explanation Timing

The explanation serves to provide users with the knowledge necessary to finish the task efficiently. This is achieved by inferring the user's mental model during the interaction and comparing it with the robot's. Whenever a disparity between these two models is detected, we can generate explanations to encourage correction of the user's mental state.

During collaboration, we use temporal parsing to get robot mental state pg_t^r from its And-Or graph at time t. As in Section 5.3.2, user mental states $\hat{pg}_t^h$ can be inferred based on communication history and action sequences. The system generates explanations when there is a mismatch between the robot mental state and inferred human mental state: $|pg_t^r - \hat{pg}_t^h| > \epsilon$. In practice, we measure $P(\hat{pg}_t^h | D_T, G)$ for every sub-tasks at each time step based on (5.6). If the probability $P(\hat{pg}_t^h = pg_t^r | D_T, G)$ is lower than a threshold, we

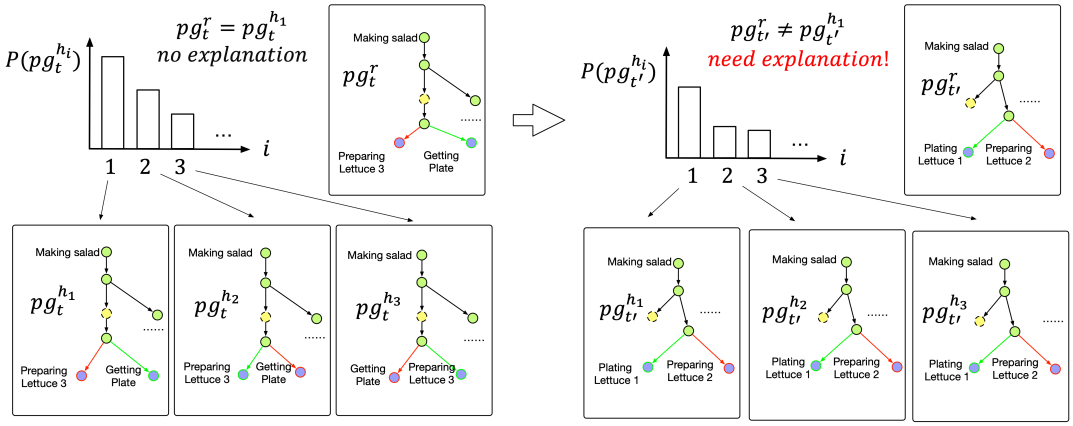


Figure 5.5: Explanation timing. At time t, sort posterior probability of $pg_t^{h_i}$ in descending order, and then compare the most possible user mental state $pg_t^{h_1}$ with robot mental state pg_t^r. Since they are the same, there is no need to explain to the user. At time t', $pg_{t'}^{h_1}$ is not equal to $pg_{t'}^r$, therefore, the robot should provide the explanation.

generate an explanation for the user. This process is shown in Figure 5.5.

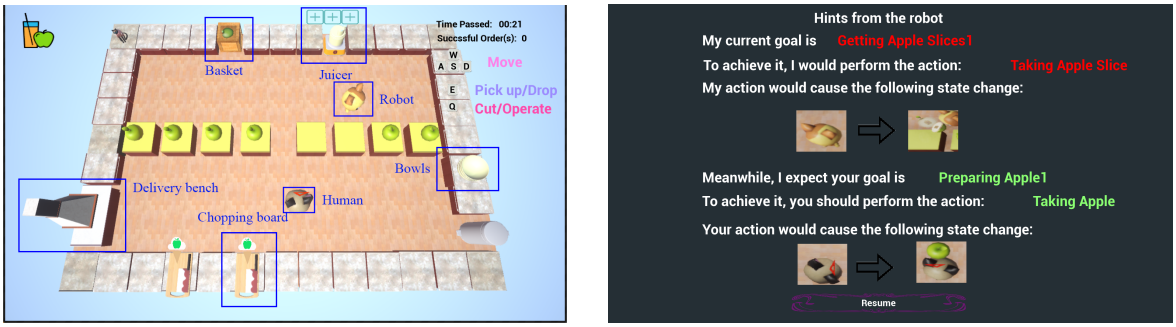
5.4.3 Explanation Content

We envision the disparity occurred between the user's mental state and robot's due to several reasons:

1. The user wants to achieve goals that are different from the robot's expectation;
2. The user performs incorrect atomic actions to achieve a sub-goal;
3. The user is unaware of the pre-condition or effect of an atomic action.

In this chapter, we do not distinguish between the possible causes of disparity when choosing the explanation timing, as they are too ambiguous. Instead, we propose to generate hierarchical explanation which consists of three components of the robot's mind representation:

1. The robot would explain the current expected sub-goals of both agents (s_t^h, s_t^r) based on its mental state pg^r; *e.g.*, "My current goal is preparing the lettuce. Meanwhile, your expected goal is getting the plate.";



(a) (b)

Figure 5.6: (a) A top-down view of our collaborative cooking game, where the user (the bottom character) collaborates with a robot (the top character) on some cooking tasks; *e.g.*, *making apple juice*. (b) The explanation interface exhibits the expected sub-tasks for both agents. Pre-conditions and post-effects of atomic actions are displayed as well.

2. The robot communicates the expected atomic actions that both agents are supposed to perform (a_t^h, a_t^r); *e.g.*, "Currently, I'm performing the action slicing the lettuce. You are supposed to perform the action taking the plate.";

3. In addition, by showing images of world states before and after an action (as shown in Figure 5.6b), the robot would also demonstrate the fluent change caused by an atomic action $f_t \xrightarrow{a_t} f_{t+1}$.

5.5 User Study

We conducted a user study in a gaming environment to evaluate our algorithm, where participants can collaborate with agents on a virtual cooking task. The gaming environment and explanation interface are displayed in Figure 5.6.

5.5.1 Experiment Domain

Our experiment domain is inspired by the video game **Overcooked**[3], where multiple agents are supposed to make use of various tools and take different roles to prepare, cook, and serve various dishes. Particularly, we use Unreal Engine 4 (UE4) to create a

[3] http://www.ghosttowngames.com/overcooked/

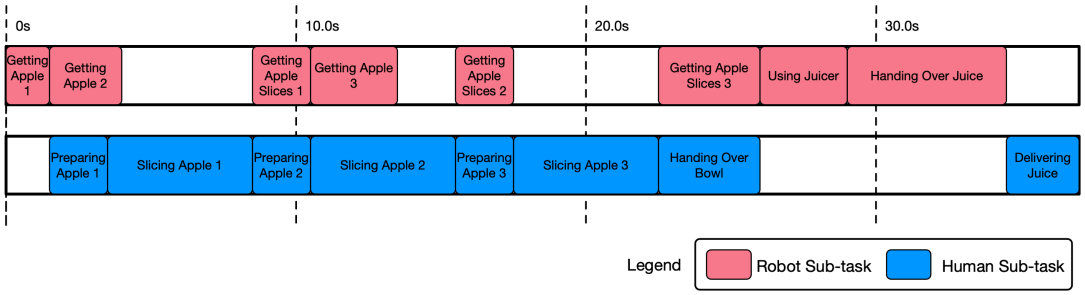


Figure 5.7: An example task schedule for *making apple juice*. The robot maintains the schedule to reflect its expectation on how the team should finish the task. Each color block represents a sub-task, performed by either robot or human. At a specific timing, we can assign tasks to both agents based on the schedule; *e.g.*, at 10.0s, the robot is *getting apple slices 1* while the user is supposed to be *preparing apple 2*. The schedule gets updated based on inferred human mental states, as shown in Algorithm 1.

real-time cooking task, namely making *apple juice*. To finish the task, teammates need to take apples from the box and slice them with a knife near the chopping board. Three apple slices should be put into the juicer before producing and delivering apple juice. Figure 5.6a shows a top-down view of the environment. The game interface is designed to be interactive (*e.g.*, object appearance will change after taking valid actions) so that people can easily play through.

To finish the task, each user needs to complete a sequence of 62 atomic actions, if acting optimally, and observe 5 different object fluent changes with a total state space around 10^9. An example task schedule is shown in Figure 5.7.

5.5.2 Experiment Design

Hypotheses: The user study tests the following hypotheses with respect to our algorithm in the collaboration:

- **H1: Task completion time.** Participants would collaborate with the robot more efficiently if the robot generates explanations based on the human mental state modeling, compared to the other conditions.

- **H2: Perception of the robot.** Participants would have higher perceived helpfulness

and efficiency of the robot, as a result of receiving explanations based on the human mental state modeling, compared to the other conditions.

Manipulated Variables: We use a between-subject design for our experiment. In particular, users are randomly assigned to one of three groups and receive different explanations from the robot:

- **Control:** Users would not get any explanations from the robot. As a result, they can learn to finish the task by interacting with the environment.
- **Heuristics:** The robot gives explanations when there is no detected user action for a period of time. This serves as a simple heuristic for the robot to infer whether the user is having difficulties in finishing the task. The timing threshold is set to 9.3 seconds, based on the result of a pre-study in which users can actively ask for explanations when they get stuck.
- **Mind modeling:** The robot gives explanations when there is a disparity between robot and human mental states.

Study Protocol: Before starting the experiment, each participant signs an informed consent form. An introduction is given afterward, including rules and basic controls of the game. As a part of the introduction, participants are given three chances to work on a simple single-agent training task, to verify their understanding. Those who fail to complete the training task in one minute would not continue the study. This is a comprehension test to exclude people who do not understand game control.

Participants who finish training get to see further instructions before starting to collaborate with the robot. They are first educated about the goal of a collaboration task (*i.e.*, making *apple juice*) and what actions the team should perform to finish it. This is done to make sure every participant has sufficient knowledge to finish the task, so that the impact of user-specific prior knowledge can be minimized. To prepare users to interact and communicate with the robot agent, we would also show them a top-down view of

the level map (as shown in Figure 5.6a), the appearance of the robot agent as well as an example of an explanation. During the task, the team is required to make and serve two orders of dishes in the virtual kitchen. At the end of the study, each participant is asked to complete a post-experiment survey to provide background information and evaluate the robot teammate.

Measurement: In the background study, we have collected from users their basic demographic information, education, as well as experience with video games.

Our objective measure is intended to evaluate the human-robot teaming performance and subjective measure is designed for evaluating users' perception of the robot. Our dependent measures are listed below:

- **Teaming performance.** We evaluate teaming performance by recording the time for the team to complete each order.
- **Perception of the robot.** We measure user's perception about the robot, in terms of its helpfulness and efficiency. Helpfulness is comprised of questions that measure users' opinion on the robot's ability to provide necessary help. Efficiency is comprised of questions that measure users' opinion on how efficiently and fluently the team is able to finish the task.

5.5.3 Results and Analysis

We recruited 29 subjects for our IRB-approved study from the university's subject pool. Most of the participants (69.3%) came from a non-STEM background. Their reported ages ranged from 17 to 36 (M=19.52, SD=2.89). All the participants have moderate experience with video games and have not played the video game **Overcooked**, which inspired our study design. Each participant got 1 course credit after completing the study. In addition, for ease of conducting the study, we discarded the data of 2 participants from the control group, as they got completely lost and failed to finish the designated task. As a result, there are 10 valid participants in the "mind modeling" and "heuristics" group,

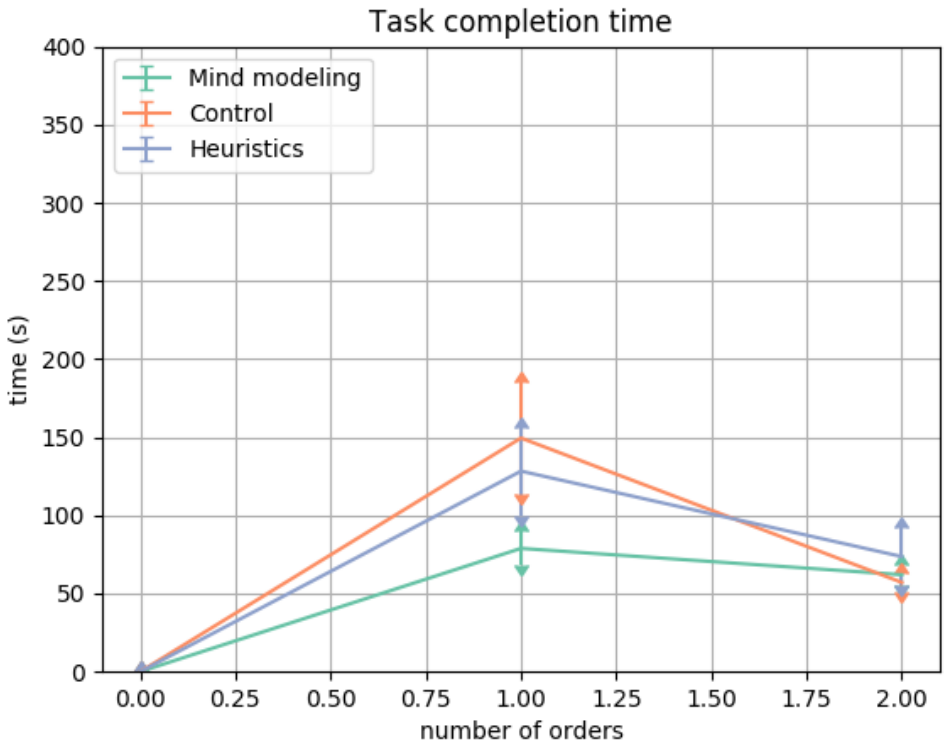


Figure 5.8: Time taken for the team to complete two orders under different testing conditions

and 7 in the "control" group.

Generally, we use ANOVA to test the effects of different experimental conditions on teaming performance and subjective perception of the robot. Tukey HSD tests are conducted on all possible pairs of experimental conditions.

As shown in Figure 5.8, we found marginally significant effects from "mind modeling" conditions on completion time of the first order ($F(2, 24) = 2.038, p = .152$). Post-hoc comparisons using the Tukey HSD tests revealed that teams could finish the first order significantly faster if users were under the "mind modeling" condition, compared to those under "control" ($p = .044$). The result is marginally significant compared to those in "heuristics" ($p = .120$), **confirming H1**. However, for the completion time of the second order, we did not find any significant effect ($F(2, 24) = 0.425, p = .658$). This is not surprising since users were asked to finish the same task twice. They could take advantage

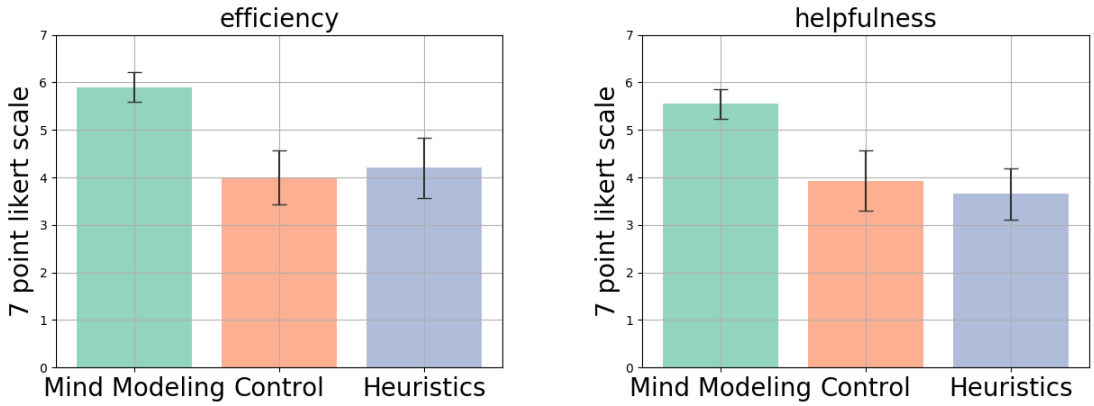


Figure 5.9: User's self-reported perception of the robot in terms of its efficiency and helpfulness.

of their previous experience working with the robot for the second order. Intuitively, the quantitative result showed that our explanation generation algorithm helped non-expert users to finish the task efficiently on their first run, while those in the control group needed to complete the task once to be able to finish it with the same efficiency.

The factorial ANOVA also revealed a significant effect of the explanation system on the perceived helpfulness ($F(2, 24) = 4.663, p = .019$) and efficiency ($F(2, 24) = 4.136, p = .029$) of the robot (Figure 5.9). **In support of H2**, post-hoc analysis with the Tukey HSD tests showed that the robot's perceived helpfulness was significantly higher under the "mind modeling" condition, compared to "control" ($p = .023$) and "heuristics" ($p < .01$). Users under the "mind modeling" were also more likely to believe the explanation system resulted in improved collaboration efficiency, compared to "heuristics" ($p = .026$) and "control" ($p < .01$).

5.6 Summary

In this chapter, we proposed a framework that allows a robot agent to improve teaming performance by communicating compelling explanations to its non-expert human teammate. By maintaining the mental state of both agents, the robot agent successfully generates explanations when the human behavior deviates from the optimal plan. By

conducting a user study on a virtual collaborative cooking task, we demonstrated that the proposed algorithm can improve efficiency and quality of the interaction.

CHAPTER 6

LEMMA: Learning Language-Conditioned Multi-Robot Manipulation

6.1 Introduction

There is growing interest in connecting human language to robot actions, particularly in single-agent systems (Shridhar et al., 2022a; Mees et al., 2022b; Zheng et al., 2022; Anderson et al., 2018; Shridhar et al., 2023). However, there remains a research gap in enabling multi-robot systems to work together in response to language input.

Recent vision and language tasks have primarily focused on navigation and object interactions (Anderson et al., 2018; Shridhar et al., 2020a; Gao et al., 2022). However, the lack of physical manipulation in these works makes the settings oversimplified. Although some recent studies, such as (Shridhar et al., 2022a, 2023), address vision and language object manipulation in single-robot settings, the language instructions provided specify only short-term goals, neglecting long-term objectives. Huang et al. (2022b) attempt to address these limitations by exploring long-horizon planning with manipulation for individual robots. Nevertheless, there remains a need to investigate multi-robot systems capable of accomplishing a broader range of long-horizon tasks while following language instructions.

Learning policies for multi-robot systems introduces distinct challenges, including diverse capabilities arising from physical constraints such as the location and reach of different robots. Moreover, task planning heavily depends on the spatial and physical relations between the objects and robots, in addition to the geometries of the objects. To

ensure suitable task assignments, an awareness of each robot's specific physical capabilities is needed.

To tackle the language-conditioned vision-based multi-robot object manipulation problem, we have developed LEMMA, a benchmark that contains 8 types of collaborative object manipulation tasks with varying degrees of complexity. Some tasks require the robot to use tools for object-object interactions. For each task, the object poses, appearances, and robot types are randomized, requiring object affordance estimation and robot capability understanding. To enable multi-task learning, each task is paired with an expert demonstration and several language instructions specifying the task at different granularities. As a result, LEMMA introduces a diverse range of challenges in multi-robot collaboration, including physics-based object manipulation, long-horizon task planning, scheduling and allocation, robot capability and object affordance estimation, tool use, and language grounding. Each aspect poses distinct challenges and is crucial for a multi-robot system that follows human instructions to complete tasks. To evaluate existing techniques on LEMMA, we further provide several baseline methods and compare their performance to each other. We assess task performance by utilizing the latest language-conditioned policy learning models. Our results indicate that current models for language-conditioned manipulation and task planning face significant challenges in LEMMA, especially when dealing with complex human instructions.

We make the following contributions:

- We design eight novel collaborative object manipulation tasks involving robots with different physical configurations implemented in Nvidia Omniverse - Isaac Sim.
- We provide an open-source dataset comprising 6,400 expert demonstrations and natural language instructions, including human and high-level instructions.
- We implement a modular hierarchical planning approach as a baseline, which integrates language understanding, task planning, task allocation, and object manipulation.

6.2 Problem Formulation

Assume a robot system comprised of N robots that is tasked to complete a complex manipulation task, the goal of which is specified by a language instruction $x_L = \{x_l\}_{l=1}^{L}$, which is a sequence of L word tokens. The full task can be decomposed into M sub-tasks $d_M = \{d_m\}_{m=1}^{M}$. d_M represents the full task, and d_m represent each sub-task in d_M. Given the language instruction x_L, our goal is to find a valid and optimal sub-task allocation. We define q_{im} and c_{im} as the quality and cost, respectively, for allocating robot i to work on sub-task m. Then the combined utility for the sub-task is:

$$u_{im} = \begin{cases} q_{im} - c_{im}, & \text{if robot } i \text{ can execute sub-task m} \\ -\infty & \text{otherwise.} \end{cases}$$

We define the assignment of sub-task m to robot i as

$$v_{im} = \begin{cases} 1, & \text{robot } i \text{ is assigned to sub-task m} \\ 0 & \text{otherwise,} \end{cases}$$

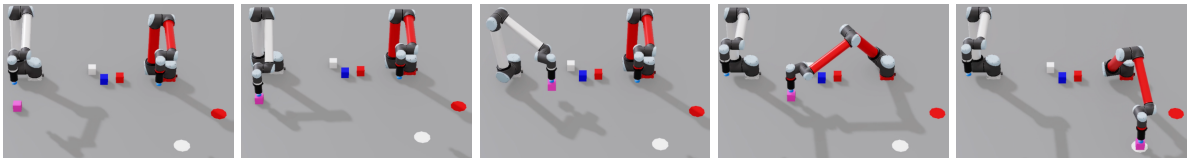
with $\gamma_m = i$ being an assignment variable for each sub-task m, indicating that sub-task m is assigned to robot i.

The goal is to maximize the utility of the full manipulation task under a time constraint. Defining the execution time for task m by robot i as τ_{im}, and the maximum time allowed to execute the task as T_{max}, we can express the task decomposition and assignment problem as follows:

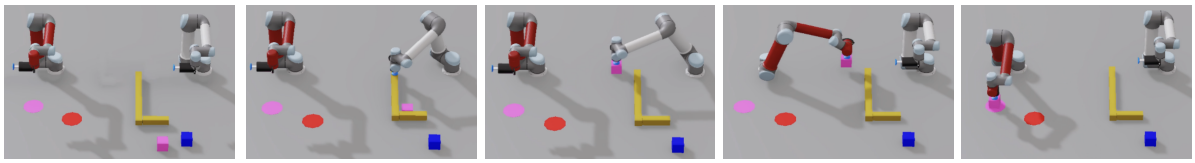
$$\arg\max_{v} \sum_{i=1}^{N} \sum_{m=1}^{M} u_{im} v_{im}, \tag{6.1}$$

subject to

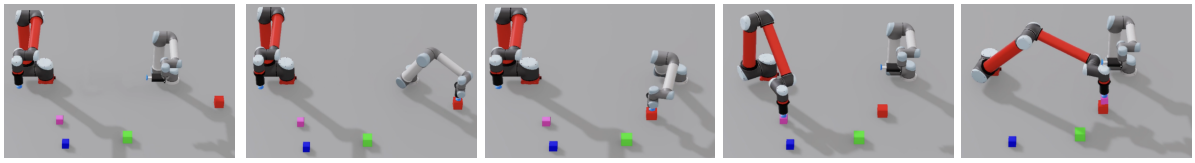
$$\begin{aligned} \textstyle\sum_i \sum_m \tau_{im} v_{im} &\leq T_{max} \\ \textstyle\sum_i v_{im} &\leq 1 \qquad\qquad\qquad \forall m \in M \\ v_{im} &\in \{0,1\} \quad \forall i \in N, \forall m \in M. \end{aligned}$$



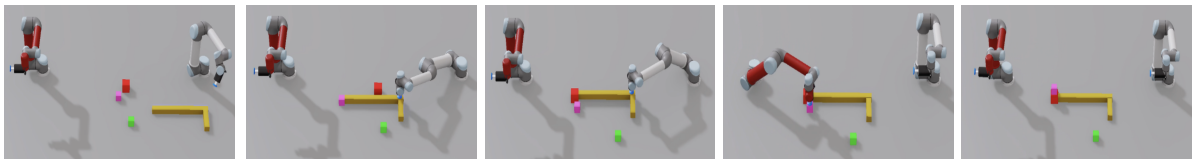
(a) Pass. Instruction: Place the pink cube on the white pad. Robots: UR10 and UR10.



(b) Hook. Instruction: Place the pink cube on the pink pad. Robots: UR5 and UR5.



(c) Stack. Instruction: Place the pink cube on top of the red cube. Robots: UR5 and UR10.



(d) Poke2stack. Instruction: Place the pink cube on top of the red cube. Robots: UR5 and UR5.

Figure 6.1: Expert demonstrations and high-level instructions of tasks in `LEMMA`. We use minimal instructions that specify the goal, thus, it is possible that two different tasks may have the same instruction; *e.g.*, tasks in (c) and (d) have the same instruction, but (d) requires robots to make use of the tool to poke the blocks so that the red robot can reach them. Note that the pair of robots involved in each demonstration can be different. The pair of robots in homogeneous settings (*e.g.*, a, b, and d) have the same reach, while in heterogenous cases the reach can be different for each robot (*e.g.*, c).

As pointed out by Korsah et al. (2013), this problem cannot be solved in polynomial time. In this work, we tackle this problem by learning from expert demonstrations, so that each sub-task can be assigned to a capable robot to ensure successful task execution. With the sub-task d_m and its assignment γ_m, an object manipulation policy $\pi(s^{t+1}|s^t, o_N^t, x_L, d_m, \gamma_m)$ can be used to move a specific object from its current pose s^t to its target pose s^{t+1}, given the observations $o_N^t = \{o_i^t\}_{i=1}^N$ and language instruction x_L. In this work, the observation O_N consists of robot joint configurations, RGBD images associated with each robot, and camera parameters.

6.3 LEMMA Benchmark

We introduce LEMMA to address the language-conditioned multi-robot manipulation problem. This benchmark is designed to evaluate a system's ability to dynamically perform task allocation and object manipulation in a tabletop environment. LEMMA sets itself apart from existing language-conditioned robotic manipulation benchmarks, such as (Zeng et al., 2021; Zheng et al., 2022; Mees et al., 2022b), in several key aspects:

- All tasks in LEMMA feature strong temporal dependencies, making the execution order of sub-tasks critically important. Out-of-order execution will result in task failure.
- LEMMA tasks are exclusively multi-agent based. Due to the robots' reachable space limitations, it is impossible to complete tasks in LEMMA using a single agent.
- As illustrated in Figure 6.1, robots are provided with only minimal high-level instructions. This requires the model to have a deep understanding of the environment and plan a sequence of actions accordingly to reach the goal specified by the instruction. Using a language classifier to determine the task type and a template to form task plans, as in (Min et al., 2022), is inadequate, as multiple tasks may share the same language instructions.
- LEMMA allows robots with different physical configurations to collaborate on manipulation tasks. Two types of robots are provided in LEMMA, namely UR10 and UR5.

A detailed comparison between LEMMA and other related benchmarks is shown in Table 2.3.

6.3.1 Task Settings

In LEMMA, we focus on tasks that require a multi-robot system to complete, given a single instruction and visual observations from top-down cameras placed above each robot. Specifically, we consider a two-robot setting under centralized control. The high-level goal is specified by a single natural language instruction such as "Place the red block on top of the white pad". However, this instruction may not provide all the necessary details on how

Task Type	Pass	Pass2	Stack	Stack2	Poke	Poke&Stack	Hook	Hook&Stack
Objects Categories	Cube, Pad	Cube, Pad	Cube	Cube	Cube, Pad, Stick	Cube, Stick	Cube, Pad, Stick	Cube, Stick
Using Tools	✗	✗	✗	✗	✓	✓	✓	✓
Passing Tools	✗	✗	✗	✗	✗	✗	✗	✓
Number of Objects	3-6	6-8	2-5	4-6	4-6	3-5	3-5	3-6
Number of Sub-tasks	2	4	2	4	3	5	4	7

Table 6.1: There are 8 types of tasks in LEMMA. Tasks require 2-7 sub-tasks, with each sub-task requiring the robot to pick up an object, move to a location and put it down. Some tasks require the robots to use tools. In addition, the most difficult task, Hook&Stack, requires passing the tool from one robot to the other.

to complete the task. To thoroughly assess system performance in language-conditioned multi-robot collaboration, we have designed 8 tasks of varying difficulty. The tasks are implemented in a simulated tabletop environment in NVIDIA Omniverse Isaac-Sim. Task statistics can be found in Table 6.1.

Pass: The first robot is required to pick up a cube of a designated color in its own reachable workspace and place it within a shared workspace; *i.e.*, a space that is reachable by both robots. Following this, the second robot must pick up the designated cube and position it on top of a specified pad in its reachable workspace.

Stack: The first robot picks up a designated cube in its reachable workspace and places it in the shared workspace. Subsequently, the second robot picks up a different designated cube and stacks it on top of the first cube.

Poke: Initially, the cube is unreachable by either robot. One robot has access to a tool and must use it to poke the designated cube into the other robot's reachable workspace. The second robot then picks up the cube and places it in the target pad in its reachable workspace.

Hook: Initially, the cube is unreachable by either robot. However, one of the robots can use a tool to hook the cube into its own reachable space. After that, the robot need to move it to the shared workspace so that the other robot can pick it up and position it

on the target pad.

Pass2: As an extension of Pass, the task requires the robots to pass two different objects to each other and place them in their respective target locations.

Stack2: As an extension of Stack, the task requires the robots to construct two block towers, each requires two cubes of different designated colors.

Poke&Stack: The task requires one robot to use a tool to poke two designated cubes into the other robot's workspace. The other robot then places one cube on top of the other.

Hook&Stack: The task requires one robot to use a tool to hook a designated cube into its own reachable space and pass the tool to the second robot, allowing it to perform its hook action on the second cube. The first robot then moves the first hooked cube to the shared workspace so that the second robot can pick it up and stack it on the second block.

6.3.2 Action Space and Observation Space

The default action space of each robot is the end-effector position. Nevertheless, alternative control mechanisms such as joint positions, joint velocities, and joint torques can also be accommodated for controlling 6 degrees-of-freedom robots. Due to physical constraints, each robot has a unique reachable workspace, allowing it to only interact with a limited set of objects in the task.

To obtain visual observations, we place a fixed RGBD camera above each robot. Each camera has a limited field of view and cannot capture all the objects. We follow Shridhar et al. (2022a) to process the vision input: we first generate the scene point cloud based on all RGBD images and camera parameters, and use the point cloud to obtain the top-down orthographic RGBD reconstruction of the scene.

6.3.3 Task Generation

For diversity among the tasks, we use a rejection sampling mechanism to generate task instances. Each task instance specifies the initial environment configurations and goal conditions. To ensure that the task completion requires both robots, we make sure that the initial location of the target object falls into the reachable workspace of only one robot. In addition to the target object, we also add some distractor objects to make the task more challenging:

1. Specify each robot's type, location, and color. There are two robot types, UR5 and UR10, respectively. Each robot has two different colors, red and white. Colors and types are randomly assigned to each robot.

2. Sample the goal condition of the task, including the colors of the target objects and their goal locations, according to the task type. *e.g.*, the goal condition for the *Stack pink on red* task is satisfied when a pink block is on top of a red block, as shown in Figure 6.1c. There are five available colors for blocks and pads, including pink, red, white, blue, and green. The color of the tool is always yellow.

3. Sample the initial locations of target objects, so that each object is only within the reachable space of a single robot.

4. Sample the colors and locations of distractor objects while making sure the colors of the target objects are unique.

The above sampling mechanism is repeated until one valid task instance is generated.

6.3.4 Expert Demonstration

Given a task specification, we use an oracle task and motion planner to create expert demonstrations. Based on the initial configurations and goal conditions of the task, the oracle creates a task plan consisting of a sequence of sub-tasks d_M, and the allocated robot γ_m for each sub-task d_m. The sub-task follows a pick-and-place procedure, specifying the

target object to pick up and the target pose at which to place the object. Once the task plan is generated, an RMPflow motion planner is utilized to generate a motion plan based on ground truth object locations at each time step. All the motion plans are executed by the designated robot, and the corresponding RGBD images and camera poses are recorded to form the expert demonstration data.

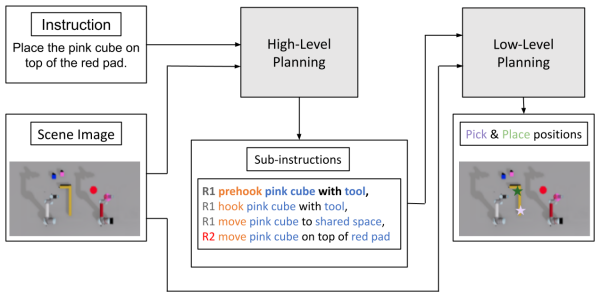
6.3.5 Language Instructions

We assign high-level instruction and a human instruction to each task instance. For high-level instruction, we manually define a template for each task and lexicalize the template using the goal of the task. For human instruction, we first crowd-sourced 500 templates on Amazon Mechanical Turk and selected 80 valid templates that are not too verbose and uniquely specify the goal given the visual observation, 10 for each task type. For each task instance, we then sample a template from the template pool and lexicalize it to generate human instruction. Some examples of the instruction templates can be found in Table 6.4.

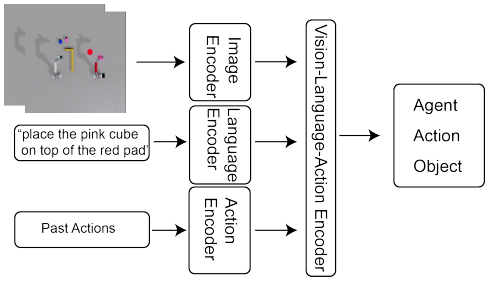
In summary, each datapoint in `LEMMA` contains the following information: 1) a manipulation task, specified by the initial configurations and goal conditions, 2) an expert demonstration, including camera poses and RGBD images at each timestep, 3) a high-level instruction and a human instruction specifying the task. As a result, we generate 800 data sessions for each task type, with a total of 6400 sessions. For each task type, we keep 700 sessions in the *training* set, 40 in the *validation* set, and 60 in the *test* set.

6.3.6 Evaluation Metrics

We have adopted success rate as the evaluation metric in `LEMMA`. Task success is defined as 1 if the task goal-conditions are met at the end of the episode, and 0 otherwise. The time constraint of each episode T_{max} is set to 100 seconds. The task specified by the instruction *Place the red block on the green circle*, for example, is considered successful if, at the end of the episode, the red block is on top of the green circular pad.



(a) Overall architecture of the baseline model.



(b) The architecture of the Episodic Transformer model used as the high-level planner.

Figure 6.2: Our baseline model involves a high-level task planning module and a low-level planning module. The high-level planning module takes as input the top-down fused scene image and human instruction to generate sub-instructions that assign the corresponding sub-tasks to robots based on their limitations. Each sub-task is specified by action primitives (in orange) and objects (in blue). The low-level planning module uses the sub-instruction and top-down projection of the scene to generate pick and place locations (visualized as purple and green stars respectively) of the gripper in the scene.

6.4 Baseline Models

Consider the problem of learning multi-agent task and motion planning with two robots given a single language instruction and visual observations. The problem can be decomposed into two sub-problems: (a) multi-robot task planning and task allocation, and (b) single-robot planning given the assigned sub-task. To this end, we design a modular baseline model which includes a high-level planner for deciding which sub-task a robot should work on and a low-level planner for generating pick and place locations for the gripper given the assigned sub-task. Figure 6.2a shows the overall architecture of our baseline model. We follow Shridhar et al. (2022a) to use the fused point clouds generated from the RGB and depth images of two cameras. Then we use the top orthographic projection of the point cloud as the visual input.

6.4.1 Action Primitives

We define six action primitives: *move, prehook, hook, prepoke, poke* and *stop*. The non-stop action primitives follow generalized pick-and-place settings. *Move* is the standard

pick-and-place primitive, requiring the robot to pick up the object and move it to a specified location. *Prehook* and *prepoke* require the robot to pick up the tool and align it with the target object to prepare for *hook* and *poke* respectively. The required height of the gripper is different for each primitive. For *hook* and *poke*, the height of the gripper after picking is set at 5cm, enabling the tool to come into contact with the target object. For other primitives, the height of the gripper is set to 30cm to avoid contact with other objects between the pick and place actions. The sequence of action primitives required to complete the task depends on the relative poses of target objects, such as cubes, tools, and pads. As a result, the same language instruction can correspond to completely different sequences of primitive actions, depending on the specific arrangement of these objects.

6.4.2 Multi-Robot Task Planning and Task Allocations

The multi-robot collaboration task can be decomposed into a sequence of sub-tasks, each can be completed by a single robot. The sub-task allocation can be represented by a tuple $(d_m^t, g_m^t, \gamma_m^t)$. $g_m^t = (e, p, q)$ represents the entities to specify the sub-task d_m^t, including the action primitive e, the pick entity p and place entity q; *e.g.*, (*Move*, *Red Cube*, *Shared Space)* indicates moving a red cube on top of the shared workspace between two robots. γ_m^t is the sub-task assignment indicating which robot is to perform the task. $(d_m^t, g_m^t, \gamma_m^t)$ are further lexicalized using templates to form the sub-instruction specifying the sub-task and its allocation (Figure 6.2a). In practice, we use Episodic Transformer (Pashevich et al., 2021), a vision and language task planner to generate the sub-instructions. The approach is a language-conditioned visual task planning method that employs a transformer architecture for long-horizon planning. As shown in Figure 6.2b, ET uses the historical visual and language information in the entire episode to capture long-term dependencies between actions. It leverages the transformer architecture to first separately encodes image histories, language instructions, and past action histories, and then perform cross-attention across modalities to decode robot assignment, action primitives, and the target object separately.

The choice between neural-based open-loop planning and closed-loop planning is often debatable. Open-loop planning cannot adapt to errors made in the planning process. However, it is more stable to train since the training distribution is often more aligned with the testing distribution. Here we consider both open-loop planning and close-loop planning for our benchmark and compare their performance. Note the original Episodic Transformer is closed-loop only and requires new observation at each time step to plan the next action (*i.e.*, *Single-step*). We modify the algorithm to use only the initial visual observation by providing it as input repeatedly during the loop to plan the whole sub-instruction sequences (*i.e.*, *Multi-step*).

6.4.3 Single Agent Object Manipulation and Grounding

In this work, we use CLIPort (Shridhar et al., 2022a) as the low-level planner. Formally, at each time step, the algorithm focuses on learning a goal-conditioned policy π that produces actions $\mathbf{a}^t$ based on the current visual observation o^t and a language instruction $x_L^t = \{x_l^t\}_{l=1}^L$. The visual observation o^t is an orthographic top-down RGB-D reconstruction of the scene, where each pixel corresponds to a point in 3D space. As shown in Figure 6.2a, in our use case, each input language instruction $x_L^t = (g_m^t, \gamma_m^t)$ specifies a sub-task being allocated to robot γ_m^t at time step t. As a result, the goal-conditioned policy is defined as

$$\pi\left(o_N^t, x_L^t\right) = \pi\left(o_N^t, g_m^t, \gamma_m^t\right) \rightarrow \mathbf{a}^t = (\mathcal{T}_{\text{pick}}, \mathcal{T}_{\text{place}}) \in A,$$

where the actions $\mathbf{a} = (\mathcal{T}\text{pick}, \mathcal{T}\text{place})$ denote the end-effector poses for picking and placing respectively. CLIPort is designed for tabletop tasks, with $\mathcal{T}\text{pick}, \mathcal{T}\text{place} \in \mathbf{SE}(2)$.

6.4.4 Multi-Agent Cliport

Since the sub-instruction already contains the sub-task allocation γ_m^t and action primitive e, the CLIPort module used as our low-level planner only needs to predict the pick and place location. To compare with this modular approach, we present a modified version of the original CLIPort module (*i.e.*, M-CLIPort) to perform task planning and

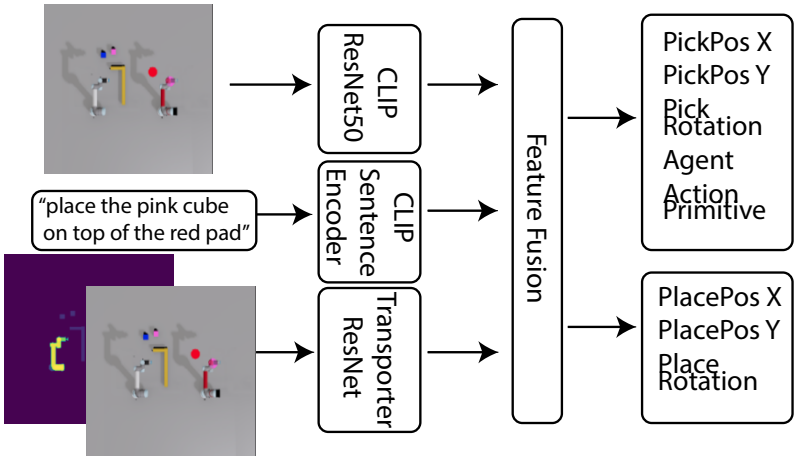


Figure 6.3: Multi-Agent Cliport Model architecture. The output is extended to include robot assignment and action primitive for each predicted action.

task allocation explicitly in an end-to-end fashion. We extend the output to a higher dimensional vector to predict the robot assignment and action primitive to use in addition to pick and place locations, as shown in Figure 6.3.

6.5 Experiments

6.5.1 Evaluation Workflow

We train the CLIPort module for 300K steps on the training set and save a checkpoint every 20K steps. Then we perform checkpoint selection on the validation split for both the high-level planner and the low-level planner. For the low-level planner, we use the ground-truth sub-instructions as input for checkpoint selection. For the high-level planner, we train the Episodic Transformer model for 30 epochs and save a checkpoint at the end of each epoch. We choose the best checkpoint based on the accuracy of predicted task plans on the validation split. We report the performance of a combination of best-performing high-level and low-level checkpoints on the test set. Since the physics and rendering in Isaac-sim are not deterministic, we evaluate all tasks for 10 runs and report the means and standard deviations.

Task Type	Pass	Pass2	Stack	Stack2	Poke	Poke&Stack	Hook	Hook&Stack	Avg
M-CLIPort	34.33±2.19	0.00±0.00	10.67±0.82	0.00±0.00	15.33±4.64	6.00±0.82	21.67±4.34	1.67±1.50	11.21±0.76
Single-step	87.00±0.67	27.00±1.25	41.67±0.00	30.00±3.86	28.00±3.86	10.33±1.24	86.70±2.36	35.00±2.36	43.21±0.45
+ GTA	**100.00±0.00**	**84.67±2.67**	**91.33±0.67**	72.00±0.67	46.67±0.00	32.33±1.70	**98.33±0.11**	**73.33±2.36**	**74.83±0.44**
Multi-step	83.06±1.15	28.06±1.78	28.89±0.79	31.11±2.48	30.83±2.31	9.17±1.27	82.78±1.57	24.72±4.66	39.83±0.84
+ GTA	**100.00±0.00**	83.89±3.00	90.00±0.00	**74.72±1.78**	**55.28±0.62**	**35.56±1.24**	96.94±0.62	58.34±1.93	74.34±0.56

Table 6.2: Performance on the test set with **high-level** instructions. M-CLIPort: Multi-agent Cliport. Single-step: high-level planning generates each sub-instruction based on the new observation. Multi-step: high-level planning generates the complete sub-instruction sequences from the initial observation. GTA: replacing the robot task allocation results from either single-step or multi-step planning by the ground truth while preserving the predicted action primitives and objects.

Task Type	Pass	Pass2	Stack	Stack2	Poke	Poke&Stack	Hook	Hook&Stack	Avg
M-CLIPort	25.83±1.86	0.00±0.00	20.00±2.36	0.00±0.00	4.58±1.38	1.25±1.38	3.75±0.72	0.00±0.00	6.93±0.47
Single-step	44.44±2.29	1.67±0.00	21.67±1.36	0.00±0.00	13.88±1.24	2.50±0.83	25.56±1.57	9.17±1.60	14.86±0.26
+ GTA	63.33±0.00	**5.56±1.24**	59.44±1.24	**1.39±0.62**	**25.28±0.62**	**9.17±1.27**	**30.00±0.00**	**19.72±0.62**	26.74±0.20
Multi-step	58.33±1.18	0.00±0.00	38.54±0.99	0.00±0.00	7.71±2.92	2.92±1.10	21.46±0.99	3.96±1.16	16.61±0.39
+ GTA	**81.89±0.55**	5.00±1.44	**70.42±0.72**	0.00±0.00	15.00±0.00	8.33±1.67	23.33±0.00	14.17±1.87	**27.27±0.30**

Table 6.3: Performance on the test set with **human** instructions.

6.5.2 Experiment Results

6.5.2.1 High-Level Instructions

The results shown in Table 6.2 compare language-conditioned policies with different task-planning modules. The M-CLIPort fails for all tasks with longer horizons. In comparison, our modular hierarchical planning approach works reasonably well for most tasks but fails for very long-horizon tasks (*i.e.*, poke&stack and hook&stack). As an ablation, we further supply ground truth task allocation to the planning module (*i.e.*, GTA). The results show that system performance can be greatly improved with ground truth task allocation. This indicates that task allocation is quite challenging since it requires understanding the reachable workspace of robots to determine which robot should be assigned to a certain sub-task.

6.5.2.2 Human Instructions

The results shown in Table 6.3 demonstrate the system performance under human instructions. Human instructions are more complex than high-level instructions since

Task Type	High-Level Instruction	Human Instruction
Pass	place the *pick-color* cube on top of the *place-color* pad	pick a *pick-color* cube from one side, put it at the center and put it on the *place-color* circle
Pass2	place the *pick-color1* cube on top of the *place-color1* pad and place the *pick-color2* cube on top of the *place-color2* pad	place the *pick-color1* cube on the *place-color1* pad and *pick-color2* cube on the *place-color2* pad in the opposite direction
Stack	place the *pick-color* cube on top of the *place-color* cube	take *place-color* block and place it in center under the *pick-color* block
Stack2	place the *pick-color1* cube on top of the *place-color1* cube and place the *pick-color2* cube on top of the *place-color2* cube	assemble two block towers by placing the *place-color1* cube under the *pick-color1* cube and the *place-color2* cube under the *pick-color2* cube
Poke	place the *pick-color* cube on top of the *place-color* pad	use the L object to push the *pick-color* block to the other robot and place it on the *place-color* pad
Poke&Stack	place the *pick-color* cube on top of the *place-color* cube	grab the brown L to push the *pick-color* and *place-color* cubes closer to the other robot, so it can grab and place the *pick-color* cube on top of the *place-color* cube
Hook	place the *pick-color* cube on top of the *place-color* pad	fetch the *pick-color* cube using the L shaped object and place it on top of the *place-color* pad
Hook&Stack	place the *pick-color* cube on top of the *place-color* cube	use the grippers to pick up the *pick-color* block from the hook and stack it on the *place-color* block

Table 6.4: Example high-level instruction and crowd-sourced human language instruction templates to specify manipulation tasks in `LEMMA`. *pick-color* and *place-color* represent the color of objects being picked up and placed on.

they feature different input lengths, levels of detail, and word choices. Our results show there is a significant gap between the performance of high-level instructions and human instructions, showing that current models are not capable of handling the increased complexity in language. Among all the results, models perform significantly worse on Stack2 and Pass2 with human instructions. This discrepancy is likely due to the fact that the orders of objects vary in human instructions for these tasks (*e.g.*, "Place the red cube under the white cube" vs "Place the white cube on top of the red cube"), as demonstrated in Table 6.4, which poses greater challenges in language understanding. In addition, for tasks requiring tool use, human instructions do not always involve the tools; *e.g.*, one instruction for poke is "push the red block toward the other robot and put it on the blue circle". In these cases, the model often fails to predict the correct object to manipulate.

6.5.2.3 Further Analysis

To provide more insight into the factors affecting task performance, we further show a breakdown of performance under different settings, including robot types and the number of distractors in the scene. We observe that in general, the task performance decreases

No. Distractors	0	1	2
Pass	**91.67±0.00**	87.06±2.35	81.18±3.50
Pass-human	**75.00±0.00**	53.92±2.19	37.25±5.55
Stack	**46.15±0.00**	40.00±3.08	16.92±2.86
Stack-human	16.67±2.87	**28.33±2.36**	15.38±6.28
Poke	**33.08±7.14**	30.00±4.44	17.50±2.50
Poke-human	**16.03±2.64**	12.96±2.62	11.46±6.67
Hook	**91.30±3.89**	83.16±3.94	84.44±7.37
Hook-human	26.09±2.51	**32.46±3.62**	17.59±2.07

Table 6.5: Impact of distractors (single-step planning)

Robot Type	UR5&UR5	UR5&UR10	UR10&UR10
Multi-step	38.60±3.25	36.32±0.47	**46.89±1.69**
Multi-step-human	15.71±0.59	**19.16±0.83**	13.06±0.53
Single-step	37.44±1.00	41.05±1.30	**51.94±1.17**
Single-step-human	**16.81±0.40**	15.57±0.33	11.94±0.96
M-CLIPort	**13.50±0.84**	11.27±0.51	9.10±2.23
M-CLIPort-human	6.62±0.93	**8.52±1.25**	4.48±1.06

Table 6.6: Impact of robot types

with an increasing number of distractor objects (Table 6.5). As for the robot types, the collaborations among two UR10s exhibit significantly higher performance using the hierarchical planning model given high-level instructions (Table 6.6). This is probably due to the fact that UR5s have a smaller reachable space compared to UR10s, making it more critical to accurately predict the reachable workspace of each robot.

6.6 Summary

In this chapter, we presented `LEMMA`, the first public benchmark for language-conditioned multi-robot tabletop manipulation. `LEMMA` combines the problems of language grounding, task planning, task allocation, tool use, capability estimation, long-horizon manipulation, and multi-modal scene understanding. All these subproblems of `LEMMA` pose significant challenges for existing algorithms.

CHAPTER 7

MindAgent: Emergent Gaming Interaction

7.1 Introduction

Large foundation Models (LFMs) have been driving the effort to develop general intelligent machines (Bubeck et al., 2023; Mirchandani et al., 2023). Although they are trained using large text corpora, their superior problem-solving capacity is not limited to canonical language processing domains. LFMs can potentially tackle complex tasks that were previously presumed exclusive to human experts or domain-specific algorithms. Recent research has shown the possibility of using LFMs to generate complex plans for robots and game AI (Liang et al., 2022; Wang et al., 2023b,a; Yao et al., 2023), marking an important milestone for LFMs as general-purpose intelligent agents. In this chapter, we investigate the planning capacity of LFMs in the context of multi-agent systems (Stone and Veloso, 2000). Compared to planning for a single agent, which has been studied extensively (Wang et al., 2023b,a), multi-agent planning imposes much higher problem-solving complexity due to an action space that grows exponentially with respect to the number of agents. The planner must simultaneously control multiple agents, avoid possible conflicts, and coordinate agents into achieving a shared goal that requires potentially sophisticated collaboration. To understand to what extent LFMs can acquire multi-agent planning skills, we first develop a new benchmark, *CuisineWorld*, which is illustrated in Figure 1.2.

To incorporate agent AIs into video games, we design *MindAgent*, an infrastructure inspired by multi-agent task allocation optimization theories, to facilitate the multi-agent planning capabilities of LFMs. Our infrastructure enables LFMs to perform complex coordination and scheduling of multiple agents in order to achieve task completion. We

conduct comprehensive evaluations with recently introduced LFMs, including GPT-4, Claude, and LLaMA, playing our `CuisineWorld` game within our `MindAgent` interactive multi-agent planning framework, leading to the following key observations:

1. **Zero shot multi-agent planning**: Powerful pretrained LFMs like GPT-4 are capable of scheduling multiple agents (ranging from 2 to 4) to complete dishes, even by collaborating with human players, by merely reading game instructions and recipes;

2. **Planning with advanced prompting**: We can significantly boost multi-agent planning performance by leveraging an emergent *in-context learning* ability (Brown et al., 2020; Wei et al., 2021) by adding only a few expert demonstrations (from different games) to the prompt, explaining the rationale of certain actions as in Chain-of-Thought prompting (Wei et al., 2022b), and providing on-the-fly feedback to the LFMs during planning.

3. **Generalization**: LFMs can potentially be generalist multi-agent planners as they are able to generalize in order to coordinate a growing number of agents and perform well in new game domains such as Minecraft.

The contributions of our work are as follows:

- We develop a new gaming scenario and related benchmark based on a multi-agent virtual kitchen environment, `CuisineWorld`. It adopts a minimal text-based game format and supports planning tasks with various structures and challenges, making it an ideal test bed for the emergent multi-agent planning (*i.e.*, scheduling and coordination) capacity of LFMs.

- We introduce `MindAgent`, an infrastructure for interactive multi-agent planning with LFMs. which demonstrates the in-context learning of the multi-agent planning capacity of LFMs and offers several prompting techniques to facilitate their planning ability, including providing few-shot demonstrations, planning rationals, and environmental feedback.

- We conduct extensive evaluations of our benchmark with multiple LFMs and prompting settings. Our experimental results validate its potential in helping develop generalist multi-agent planners.
- We deploy `MindAgent` in real-world gaming scenarios and demonstrate its ability to power human-AI interactions.

Compared to canonical domain-specific automated planning systems, although multi-agent planning with LFMs is more likely to be bottlenecked by high computational cost, context length limitations, non-optimal plans, *etc.*, it can potentially improve planning performed by *in-context learning* from data without fine-tuning, seamlessly adapt to new planning problems across different domains, and offer a more flexible interface to human collaborators. Ultimately, our investigation into the leveraging of LFMs for general-purpose scheduling and coordination can elucidate how such skills may be acquired by learning from large text corpora, and is potentially instrumental to the future development of more effective LFM-based planners.

7.2 The `CuisineWorld` Game

We introduce `CuisineWorld` as a novel and flexible game for multi-agent scheduling and coordination in a *virtual kitchen* environment. In this game, a multi-agent system must supervise multiple agents and coordinate them, with the goal of completing as many dish orders as possible. The game is equipped with a textual interface since our focus is on evaluating LFM-based planning agents. Our modularized design separates tasks and game engines, allowing inclusion of more tasks (dish types) and domains ("kitchen" implementation via text-based engine, Unity, Minecraft, *etc.*).

7.2.1 Tasks and Reward

A task in `CuisineWorld` is a dish order, ranging from the most basic `tunaSashimi`, which can be made by simply chopping raw tuna meat, to sophisticated dishes like `porkPasta`

requiring various cooking tools. In a game episode with a maximum of T steps, in every *task interval* τ_{int}, a new task or dish order will be added to the active task list. A task will be regarded *completed* and be removed from the active task list when the corresponding dish has been placed on the serving table. Alternatively, a task will be deemed to have *failed* and be removed from the list after its *lifetime* τ_{lft}, which depends on the complexity of the dish, is exceeded. Along with the tasks, the game provides rewards and penalties or feedback on certain occasions, *e.g.*when a task is just completed, when infeasible commands are dispatched, *etc.*. We support five different actions 1) goto 2) get 3) put 4) activate 5) noop. The state space contains descriptions of the environment and agents. Due to space limitations, we refer the reader to additional details in Section B.4.

7.2.2 Collaboration Score (CoS)

We need to evaluate to what extent the dispatcher (played by an LFM) can coordinate multiple agents to complete dish orders across a variety of scenarios. We are particularly interested in the question: can the dispatcher continue to coordinate the agents into efficient collaborations as τ_{int} decreases; *i.e.*, as more dish orders are flooding in? Our hypothesis is that an ideal dispatcher should be capable of coordinating the agents until there are way more tasks than the system can handle. Therefore, we introduce a *collaboration score* (CoS), defined as

$$\text{CoS} = \frac{1}{M}\sum_{i=1}^{M}\frac{\#\text{ completed }\left[\tau_{\text{int},(i)}\right]}{\#\text{ completed }\left[\tau_{\text{int},(i)}\right] + \#\text{ failed }\left[\tau_{\text{int},(i)}\right]}, \tag{7.1}$$

where # denotes the number of tasks and M is the total number of τ_{int} intervals evaluated. Effectively, CoS is the average task completion rate across different τ_{int} conditions. In our default setting, we use $M = 5$. While the actual values of τ_{int} depend on the game level, we ensure that they span a wide range of difficulties including both relaxed and intense scenarios.

In summary, `CuisineWorld` is a game that emulates a virtual kitchen in which several

robotic agents are commanded to use various cooking tools and ingredients to prepare as many dish orders as possible in a limited period of time. To necessitate collaboration, new orders will keep flooding in while the existing ones should be completed before their expiration times. Therefore, LFMs must properly coordinate the agents to maximize overall productivity. `CuisineWorld` offers game levels with a wide range of planning difficulty: dishes with different complexity (number of ingredients and tools involved), number of agents, order frequency and lifetime, *etc.*., making it a useful test bed for LFM-based multi-agent planning.

7.3 `MindAgent` Gaming AI Infrastructure

Our first foray into the challenging `CuisineWorld` benchmark is an interactive multi-agent planning framework with LFMs. It facilitates in-context learning and adopts a minimalist design for the purposes of demonstrating the scheduling and coordination capacity of LFMs, while also bringing in exploratory prompting techniques that facilitate better planning and inform future approaches in this domain. Our `MindAgent` infrastructure comprises prompt, current state, and memory, as shown in Figure 7.1 with details illustrated as follows:

Prompt incorporates four distinct sub-components: recipes, general instructions, inference knowledge, and a one-shot demo.

Recipes outline hierarchical procedures for preparing various dishes at a given level. They specify the ingredients necessary for each intermediate or final product, the appropriate tools, and the outcome.

Instructions detail the foundational rules of `CuisineWorld`, delineating the array of actions agents can undertake within the game and enumerating the characteristics of every tool available. Moreover, they inform agents about the base ingredients retrievable from storage, as well as all potential intermediate products they can procure. Agents are also explicitly advised to remain cautious about feedback from the environment.

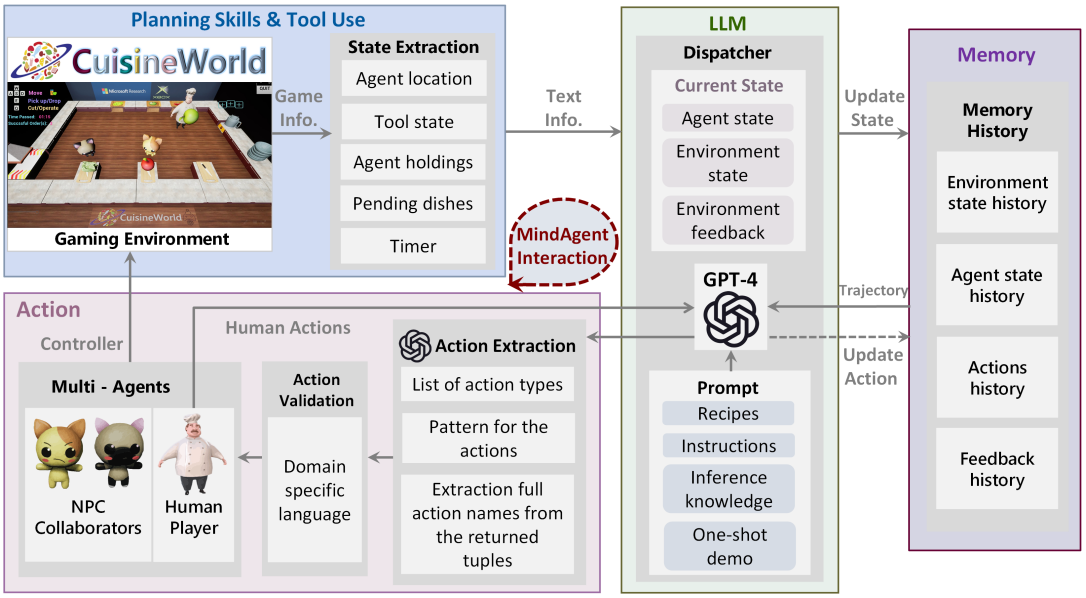


Figure 7.1: The `MindAgent` Infrastructure. **Planning Skill and Tool Use:** The game environment requires diverse planning skills and tool use to complete tasks. It generates relevant game information and converts the game data into a structured text format that the LFMs can process. **LFM:** The main workhorse of our infrastructure makes decisions, thus serving as a dispatcher for the multi-agent system. **Memory History:** A storage utility for relevant information. **Action Module:** Extracts actions from text inputs and convertd them into domain-specific language and validates DSLs so that they cause no errors during execution.

Inference Knowledge encapsulates insights and helpful hints for the agent, which when utilized appropriately can guide agents to sidestep potential errors and improve their collaborative efficiency.

One-shot Demo presents a step-by-step demonstration of the preparation of a distinct dish, different from other dishes at the current level, spanning several time steps, each of which is incorporated as part of the prompt. The demonstration illustrates the major procedures for cooking a dish in `CuisineWorld`, including obtaining ingredients, putting ingredients into different tools, transporting intermediate ingredients, and delivering the final dish to the serving table.

Current State provides a snapshot of the prevailing observations from the environment. It encompasses information such as the locations of agents, the objects currently in the possession of agents, the tools that are accessible within the environment, the

ingredients present within each tool, and the tools that are actively in use. Moreover, it includes optional feedback from the environment, triggered when agent actions violate the rules of the environment; for instance, when assigning two distinct actions to the same agent.

Memory archives the history of interaction with the environment. Specifically, it chronicles the state of the environment and the state of the agents at every time step.

In addition to the prompt modules, other modules are implemented to help interface between LFMs and `CuisineWorld`, as follows:

Action Extraction employs a regular expression matching procedure to distill agent actions from the textual output of the LFMs. This module is indispensable because LFM output is not always clean, but may include information reflecting its internal thought processes or even issue apologies for prior missteps in reaction to environmental feedback.

Action Validation utilizes a look-ahead checking mechanism. This module parses the proposed actions, assessing their feasibility. If an action is deemed unexecutable, an error message is returned.

7.3.1 Infrastructure Mechanisms

Assuming a multi-agent system with N agents, the system must complete a sequence of P different tasks. Each task has M_p different sub-tasks. Furthermore, the number and types of tasks are unknown at the beginning of the episode. The environment will sample a task for the agents to finish during a given interval. The agents must complete the designated task along with other tasks in the task queue. Additionally, each task has an expiration time, after which the task will be marked as a failure. The objective of the multi-agent system is to finish as many tasks as possible and fail as few tasks as possible within a given time frame.

To find optimal task planning, scheduling, and allocations. We define q_{pim} and c_{pim} as quality and cost, respectively, in the context of allocating agent i to work on sub-task m

of task p in the episode. The combined utility for the sub-task is

$$u_{pim} = \begin{cases} q_{pim} - c_{pim}, & \text{if agent } i \text{ can execute sub-task } m \text{ of task } p \text{ in the episode;} \\ -\infty, & \text{otherwise.} \end{cases} \tag{7.2}$$

The assignment of sub-task m to agent i is

$$v_{pim} = \begin{cases} 1, & \text{if agent } i \text{ is assigned to sub-task } m \text{ of task } p \text{ in the episode;} \\ 0, & \text{otherwise.} \end{cases} \tag{7.3}$$

The goal is to maximize the utility of the episode subject to a time constraint. We define the execution time for task m by agent i for task p in the episode as τ_{pim}, and the maximum time allowed to execute the task as $T_{\max}$, we express the task decomposition and assignment problem as

$$\begin{aligned} &\arg\max_{v} \sum_{p=1}^{P} \sum_{i=1}^{N} \sum_{m=1}^{M_p} u_{pim} v_{pim}, \\ &\text{subject to} \\ &\sum_{p} \sum_{i} \sum_{m} \tau_{pim} v_{pim} \leq T_{\max}, \\ &\sum_{i} v_{pim} \leq 1 \quad \forall m \in M, \forall p \in P, \\ &v_{pim} \in \{0, 1\} \quad \forall i \in N, \forall m \in M, \forall p \in P. \end{aligned} \tag{7.4}$$

Since this problem cannot be solved in polynomial time, we tackle it by leveraging LFMs.

Our prompt design choices try to help an LFM system solve Equation 7.4. In practice, we reformulate the equation with qualities or rewards expressed in natural language as environmental feedback. For example, when the agent successfully collects an item, the environment emits a signal "collect finish". When the dispatcher assigns a different task to the same agent, the environment emits a signal "agent IDs cannot be the same". As rewards are not immediately observable, we borrow spirits from temporal difference

	very simple			simple			intermediate			advanced			Average
	level 0	level 1	level 7	level 2	level 4	level 8	level 3	level 9	level 10	level 5	level 11	level 12	
2 Agents	0.727	0.706	0.682	**0.687**	**0.664**	0.504	0.764	0.725	0.701	0.661	0.692	0.559	0.673
3 Agents	**0.781**	**0.778**	**0.780**	0.528	0.600	0.455	0.822	**0.771**	**0.815**	0.689	**0.733**	**0.570**	**0.694**
4 Agents	0.771	0.761	0.761	0.505	0.592	**0.626**	**0.848**	0.744	0.790	**0.692**	0.675	0.534	0.692

Table 7.1: Agent CoS performance scores on very simple, simple, intermediate, and advanced tasks for various numbers of agents.

learning. State-action history is accumulated into the memory history. Due to context length limits, it is infeasible to fit the entire history into the context window. We select a fixed horizon history as part of the prompt. We further express the constraints of the system in natural language and repeat important constraints multiple times if necessary.

7.4 Experiments and Results

We have conducted extensive experiments in `CuisineWorld`. We first introduce the experiment settings and then present an analysis of our empirical results. Our experiments focused on addressing the following research questions:

Q1: How efficiently can the model dispatch multiple agents?

Q2: Can the model dispatch agents for dynamic, on-the-fly goals across different tasks?

Q3: To what extent can the existing methods collaborate with human users?

Q4 What is the human perception of collaborating with numerous intelligent agents?

Q5: How do various components of the input prompt influence the model's performance?

Q6: How do other LFMs perform compared to GPT-4?

7.4.1 Experimental Regimen I: LFMs Dispatch Multi-Agent NPCs (Q1, Q2)

Figure 7.2 and Table 7.1 report the performance of our system under different settings.

As shown in Figure 7.2, in general, increasing the number of agents from 2 to 3 will increase the overall performance. However, when increasing more, the overall performance might drop due to the complexity of multi-agent collaborations, as shown by the corresponding CoS by level. As shown in the tables, the CoS is the highest when

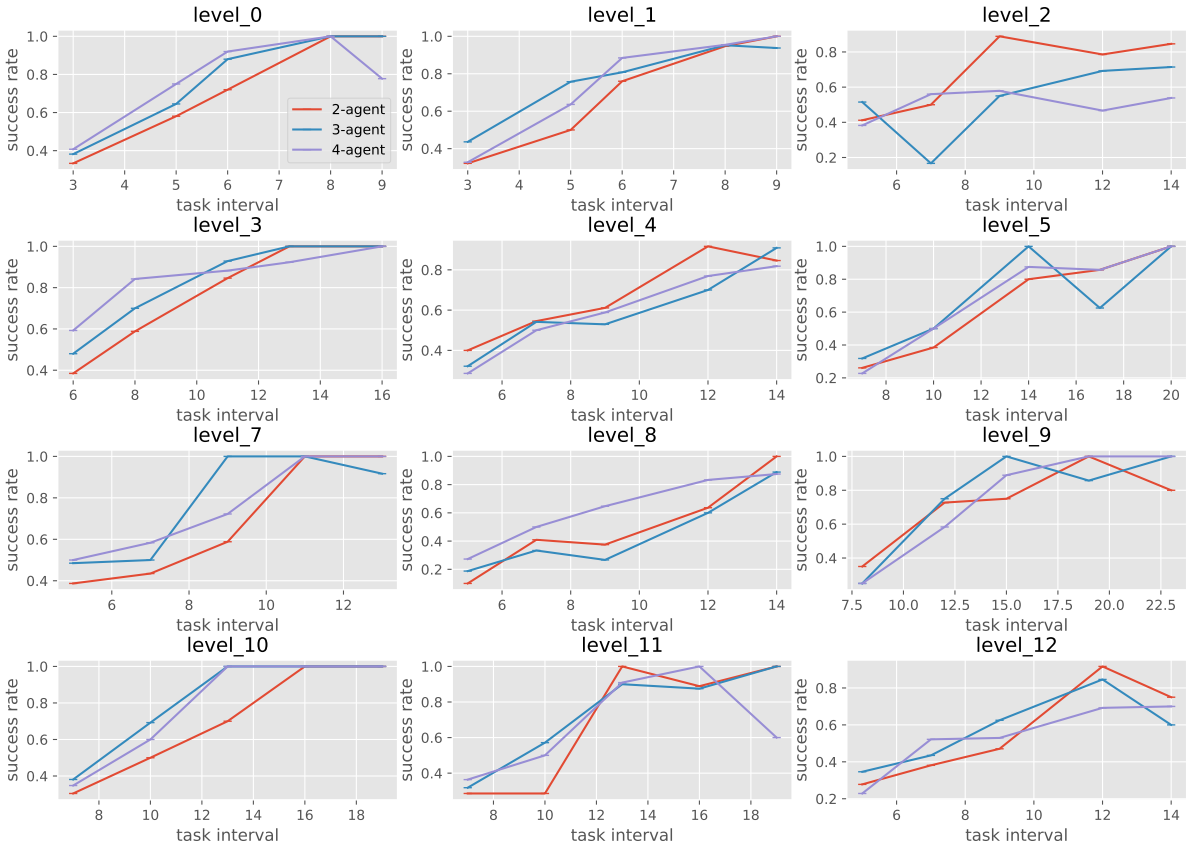


Figure 7.2: Collaboration efficiency curves on several levels.

there are two agents in two cases. The CoS is the highest when there are three agents in seven cases. The CoS is the highest when there are four agents in three cases. Second, we observe that the system performance degrades with more agents under less demanding conditions, indicating that the LFM dispatcher struggles with fewer tasks.

7.4.2 Experimental Regimen II: Human and Multi-NPCs with LFMs

We recruited 12 subjects for our internal ethics review approved study, including 2 females and 10 males. During testing, we recorded only user interaction data. Identifiable information was not recorded. Subjects could quit at any time. We used ANOVA to test the effects of different experimental conditions on collaboration performance and the subjective perceptions. Tukey HSD tests were conducted on all possible pairs of experimental conditions.

We conducted a user study in our gaming environment that addresses **Q3** and **Q4**. The user study evaluates the LFM dispatcher's ability to collaborate with humans, where participants are collaborating with 1, 2, and 3 agents or working alone on the virtual cooking tasks (level 3).

The user study tests the following hypotheses:

H1: Task productivity. Participants have higher productivity when collaborating with AI agents.

H2: Task productivity with more agents. Participants have higher productivity when collaborating with more AI agents.

H3: Perception of the AI agents. Participants have higher perceived task efficiency and more fun playing the game due to collaboration.

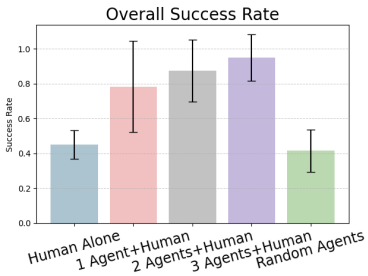
We use a within-subject design for our experiment. Every user tries to finish the task solo or collaborates with different numbers of agents with varying competency. We randomize the order of the treatment to mitigate practice, fatigue, and carryover effects.

Single agent: Participants work by themselves.

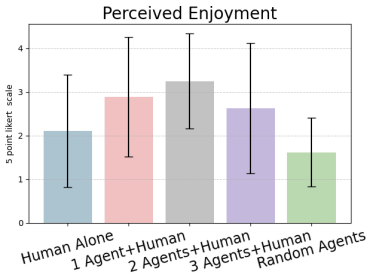
LFM-powered multi-agent system: Participants collaborate with the multi-agent AI system powered by an LFM.

Random agent: Random agents execute random actions from a pool of valid actions. Participants collaborate with random AI agents.

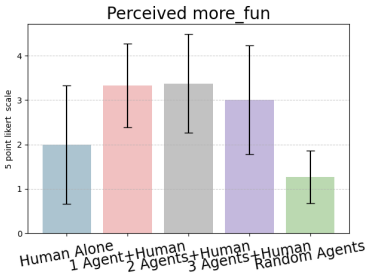
As shown in Figure 7.3 we found significant effects on the team collaboration success rate $F(4, 55) = 28.11$, $p < 0.001$. Post-hoc comparisons using Tukey HSD tests revealed that the team comprising the human player with LFM agents achieves a higher success rate than the human working alone ($p < 0.001$) across different numbers of agents, **thus confirming H1**. Although collaborating with more agents led to a greater success rate, collaborating with one agent was not significantly different from collaborating with two or three agents ($p = 0.774$ and $p = 0.231$, respectively). Therefore, we are **unable to confirm H2**. We observed that human players have more fun playing the game when collaborating with LFM-powered AI agents than when playing alone ($p = 0.0126$). Players felt that collaboration with AI agents leads to higher productivity ($p = 0.0104$), **thus**



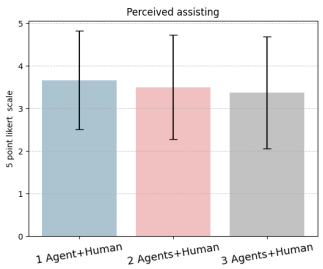
(a) **Collaboration score:** The collaboration score is higher if more agents are collaborating with human players, although the difference is not significant.



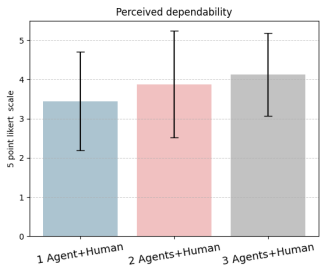
(b) **Perceived enjoyment:** Humans enjoy the game more if they collaborate with the right number of agents.



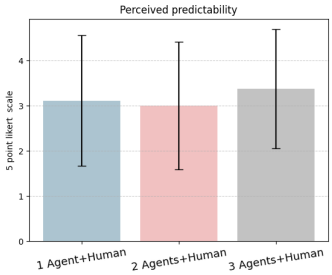
(c) **Perceived more fun:** Players enjoy the game more because of collaboration with competent agents.



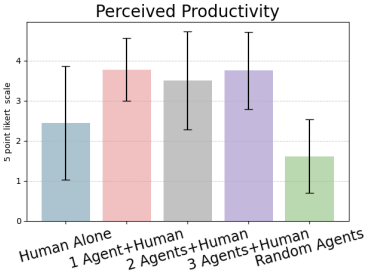
(d) **Perceived Assisting:** There is no significant difference in terms of human perceptions of helpfulness when collaborating with more agents, even though the task success rate is higher.



(e) **Perceived dependability:** When collaborating with more agents, players depend on the agents more.



(f) **Perceived Predictability:** There is no difference in terms of the predictability of agent behaviors when collaborating with more agents.



(g) **Perceived productivity:** Players think collaborating with AI agents will improve productivity.

(h) **Perceived Trust:** There is no difference in terms of trust when collaborating with more agents.

Figure 7.3: Results of human evaluations

Ablation	CoS
GPT-4 (full)	0.764
GPT-4 w/ only few-step	0.710
GPT-4 w/o inference knowledge	0.714
GPT-4 w/o feedback	0.311

Table 7.2: Additional ablation on Level 3 for 2 agents

confirming H3. Additionally, when playing with AI agents, human players take their actions based on other players' actions ($p = 0.00266$). Human players also found that AI agents are more predictable than random agents ($p < 0.001$). Further insights from player feedback highlighted an intriguing trade-off: while greater numbers of agents improved overall task success rates, this reduced the enjoyment of the game. Often, players felt sidelined and less involved. Thus, game developers should adjust AI performance to maintain player engagement and fun. As suggested by Yuan et al. (2022), aligning human values with AIs is a promising approach.

7.5 Ablation Study for Multi-Agents

7.5.1 Study of the Prompt Components (Q5)

In Table 7.2, we elucidate the performance of LFM dispatchers with certain components of the prompt omitted. We chose level 3 as the basis for our ablation study. It was selected because it displayed a clear correlation between an increased number of agents and improved performance, serving as a stable benchmark for evaluating the LFM's coordination capabilities. We recognize that the LFM may not perform consistently across all levels on the challenging CuisineWorld benchmark. When performance is variable or poor on certain levels, it can obscure the effects of ablation studies due to multiple confounding factors. Details about the prompt can be found in the appendices. Specifically, for these tests, we excluded individual components such as the inference knowledge, reduced the prompt example to a mere two steps instead of the complete demonstration, and evaluated the model without environmental feedback.

Ablation on demo	CoS
4agent using 4agent demo	0.848
4agent using 2agent demo	0.851
3agent using 3agent demo	0.822
3agent using 2agent demo	0.775

Table 7.3: Using different numbers of agents as one-shot demonstrations on Level 3

	GPT-4	Claude-2	LLaMA2	ChatGPT
2 Agents	0.686	0.3125	0	0
3 Agents	0.822	0.372	0	0
4 Agents	0.848	0.473	0	0

Table 7.4: CoS performance scores of other LFMs on Level 3

Table 7.2 indicates a significant drop in performance when environmental feedback is excluded, underscoring its pivotal role in the efficacy of the LFM dispatcher. Replaying action sequences reveals that, without feedback, the LFM dispatcher tends to repeat mistakes and gets stuck in specific states for prolonged durations. Another key takeaway is that a succinct two-step demonstration of input and output format can still achieve impressive performance for unseen tasks with dynamic objectives. Notably, in these two-step instances, there is no explicit guide to finishing any tasks, yet the model does not merely complete the task but continually performs additional tasks within the same episode. Furthermore, we observe that integrating human-crafted inference knowledge bolsters the performance of the LFM dispatcher. Lastly, even with few-shot demonstrations involving fewer agents, the LFM dispatcher retains satisfactory performance, as shown in Table 7.3.

7.5.2 Study of the Performance of Other LFMs (Q6)

To study how other LFMs perform on our tasks, we tested the collaboration performance of GPT-3.5, Claude-2, and LLaMA2, and Table 7.4 summarizes the results. For a fair comparison, all tests employed identical prompt inputs.

We observed that while other LFMs tend to underperform, models such as Claude-2 manage to complete the task to a considerable extent.

7.6 Emergent Abilities

Across our experiments, our `MindAgent` framework exhibits the following emergent properties:

7.6.1 Emergent Collaboration Task Understanding

As shown in Table 7.2, especially in the few-step ablation entries, GPT-4 exhibits its proficiency even when not provided with a full demonstration of specific tasks. To clarify, a "full few-shot demo" typically refers to a comprehensive demonstration of a task, detailing each step and procedure involved. By contrast, we provide GPT-4 with only a partial demonstration or a glimpse of the task executing only two steps. Yet, despite this limited input, GPT-4's performance is remarkable. This underscores GPT-4's impressive **emergent zero-shot multi-agent planning** abilities. Beyond simply completing unseen tasks, GPT-4 also demonstrates adaptability by dynamically prioritizing multiple different tasks as they arise, emphasizing its **emergent multi-task, on-the-fly planning** skills.

7.6.2 Emergent Multi-Agent Reasoning Abilities

Referring to Table 7.3, GPT-4 has the ability to deploy more agents based on demonstrations of fewer agents. For instance, it can effectively dispatch 4 agents having only seen demonstrations involving 2 agents. The performance is better when 4 agents use 2-agent demos compared to 4-agent demos, possibly because the task suits 2 or 4-agent teams. With 4 agents, the model can create two independent teams of two, showing its ability to allocate tasks and plan for more agents than previously experienced.

7.7 Novel Game Adaptation

In line with our ongoing efforts to create collaborative, in-game, multi-agent systems, we integrated our infrastructure into Minecraft (Figure 7.4). In this adaptation, we designed several unique cooking tasks where two in-game agents, Alex and Steve, must cook various

(a) Multi-Agent (b) Human-NPC (c) VR

Figure 7.4: (a) Alex and Steve are collaborating to kill different animals. (b) A human player instructs the agents to perform certain actions. (c) A human player collaborating with agents in VR.

GPT-4 Minecraft	$\tau_{\text{int},(1)}$	$\tau_{\text{int},(2)}$	$\tau_{\text{int},(3)}$	$\tau_{\text{int},(4)}$	$\tau_{\text{int},(5)}$	CoS
Performance	0.195	0.381	0.704	0.792	0.833	0.581

Table 7.5: Performance of the `MindAgent` framework in Minecraft

types of meat. After cooking, they must deposit the meats into a chest. See Table 7.5 for the experimental results. Refer to Section B.7.4 for the details of Minecraft actions. Section B.7.1 provides more details about transferring to Minecraft.

Incorporating game-specific domain knowledge into the system is crucial for games in which specific knowledge plays an important part. In CuisineWorld and Minecraft, we inject domain-specific knowledge (such as recipes) directly into the context, which the LFMs utilize to inform the decision-making and collaboration strategies. This demonstrates the feasibility of adapting `MindAgent` to other domains where specific knowledge plays a crucial role. In addition, techniques like Retrieval-Augmented Generation (RAG) and Fine Tuning may be pivotal in further developing `MindAgent`'s ability to handle domain-specific complexities.

7.8 Summary

We introduced `MindAgent`, an infrastructure for multi-agent collaboration through LFMs across multiple gaming domains. We investigated its multi-agent planning capabilities, and we deployed our infrastructure into real-world video games that demonstrate its multi-agent and human-AI collaboration effectiveness. Additionally, we presented

`CuisineWorld`, a text-based multi-agent collaboration benchmark that provides a new auto-metric Collaboration Score (CoS) to quantify collaboration efficiency.

CHAPTER 8

Conclusions

8.1 Summary

Interaction with the environment is the cornerstone of intelligence, as intelligence emerges in agent-environment interactions as a result of sensorimotor activities (Smith and Gasser, 2005). Directly interacting with real-world environments can be slow, expensive, and dangerous. Therefore, large-scale simulation systems will serve as the bedrock for developing future generations of intelligent agents. In this dissertation, we pursued large-scale simulations for embodied AI and robotics. Through the simulation environments and their related benchmarks and datasets, we studied several problems involving multi-modal sensory activities and concluded that current systems do not suffice to tackle the complex challenges that arise with high-fidelity simulation environments. Although challenges remain, we demonstrated in this thesis that sim2real transfer from simulation environments is feasible.

In Chapter 3, we presented an indoor task completion human simulator, where a humanoid robot must use various tools to manipulate objects, navigate around the room, and find objects to finish a series of complex cooking tasks. We further collected an indoor human activity dataset for human intention prediction. In Chapter 4, we pushed the boundaries of what is possible in robotics simulations for photo-realism and physical-realism. Rather than defining object states as discrete, we defined a continuous range of object states, connected human language to robots, and measured their grounded state understanding. Additionally, we presented a benchmark for various levels of systematic generalizations. We found that the current system cannot generalize to unseen object

states, prompting further research in this area. In addition, we conducted sim2real transfer experiments and found that sim2real transfer is possible within our framework.

The simulation environments developed in Chapter 3 and Chapter 4 allow researchers to study interactive visual task planning, scene understanding, and grounded robot manipulation. In Chapter 5, we further studied human-robot collaboration through a theory of mind framework. We demonstrated the effectiveness of mental modeling and explanations. In Chapter 6, we further studied multi-robot collaboration. We demonstrated that using language as an intermediate representation with an explicit task planning module improves downstream task performance. In Chapter 7, we studied multi-agent collaboration using LLMs. We found that LLMs can serve as an efficient multi-task, multi-agent coordinator as well as a natural interface for communicating and collaborating with human users.

8.2 Future Directions

In this thesis, we proposed several methods to scale up robot simulations and conducted preliminary sim2real transfer tests. We also conducted several human-AI collaboration experiments. However, numerous interesting problems remain. The following problems are well worth investigating in the future:

- **Simulation Systems:** Despite creating large-scale datasets, the current scope of robotics simulation systems is still modest compared to the extensive data used for training large-scale vision-language models. Further scaling of simulation systems is crucial. Additionally, the most important thing is not the scale of the dataset but rather the diversity of the dataset. Creating a large number of diverse tasks and a systematic evaluation system will be crucial for future systems.

- **Multi-Contact Dynamics:** The physics system of current robotics simulations needs refinement, particularly in simulating accurate contact dynamics for better manipulation systems. Additionally, a more systematic approach to sim2real transfer

is necessary, moving beyond human-designed heuristics towards automated methods for setting simulation parameters that closely match real-world conditions.

- **Enhancing Sensor Systems:** Current systems predominantly rely on visual sensors. Incorporating a richer array of sensors, including tactile sensors and other modalities with real-time feedback with compliance mechanisms, is essential for developing adaptive robotics systems.

- **Few-Show Generalization:** Current systems often perform poorly in novel scenarios, as highlighted in this dissertation. We hope to bridge the gap between human generalization capabilities and current AI systems. We believe that choosing the correct representation and inductive biases are the key to unlocking this potential.

- **Cross Embodiment Generalization:** Current robotics systems often learn from demonstrations of the same embodiment. However, there is a wealth of interactive data from humans and animals interacting with their environments. If we can leverage such data for efficient learning, this presents a huge opportunity to scale to unseen tasks.

- **Latency:** State-of-the-art systems require a large amount of computing power to make simple decisions. For example, Brohan et al. (2023) can only process up to 5 Hz with cloud computing. This results in significant latencies and high deployment costs. Increasing the computational speed with smaller models or different sensor modalities will be essential to facilitate natural human-robot interaction for future real-time applications.

We believe that solving the aforementioned issues requires interdisciplinary expertise in computer vision, natural language understanding, computer graphics, machine learning, robotics, neuroscience, and cognitive science. With rapidly increasing computational power and data, we believe that these challenging problems can be solved through collaboration between expert research teams with diverse technical backgrounds.

APPENDIX A

ARNOLD Benchmark Details

A.1 Environment

A.1.1 Assets in USD Format

Universal Scene Description (USD) is a format for 3D scene descriptions. The process of parsing assets into Omniverse involves the transformation of graphical data into USD files. This primarily provides developers with a streamlined method of accessing and retrieving relevant information from the assets, including scene files, articulation bodies, animations, *etc.*. By utilizing a USD file format, users can easily access a wide range of assets and subsequently inform Omniverse of the relevant content required for their application, eliminating the need for tedious, manual retrieval of information from each asset individually.

Working With the 3D-Front Dataset: The 3D-Front dataset consists of a set of professionally designed synthetic indoor scenes. It features a large number of rooms that are furnished with high-quality textured 3D models. The dataset is composed of three primary components: models, scenes, and textures. And it contains tens of thousands of room layouts with thousands of furnished objects. To parse the 3D-Front dataset into USD format, we apply the following steps as shown in Figure A.1:

- Parse the original scene files (JSON) into a data frame containing the mesh and furniture information.
- Use Maya MEL script to load scenes into Autodesk Maya.
- Apply Maya and Omniverse converter to save the scenes in USD format.

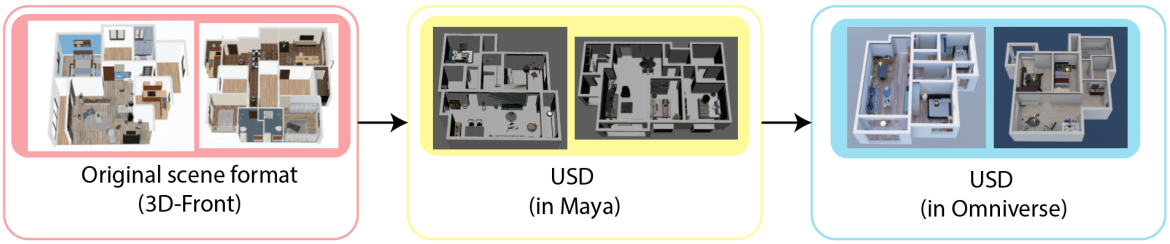


Figure A.1: Pipeline of scene parsing. After pre-processing the original 3D-Front scene data, we build an automatic pipeline to load the scene layout into Autodesk Maya with custom designs. Next, we convert the layout file to USD format and deploy it in Omniverse.

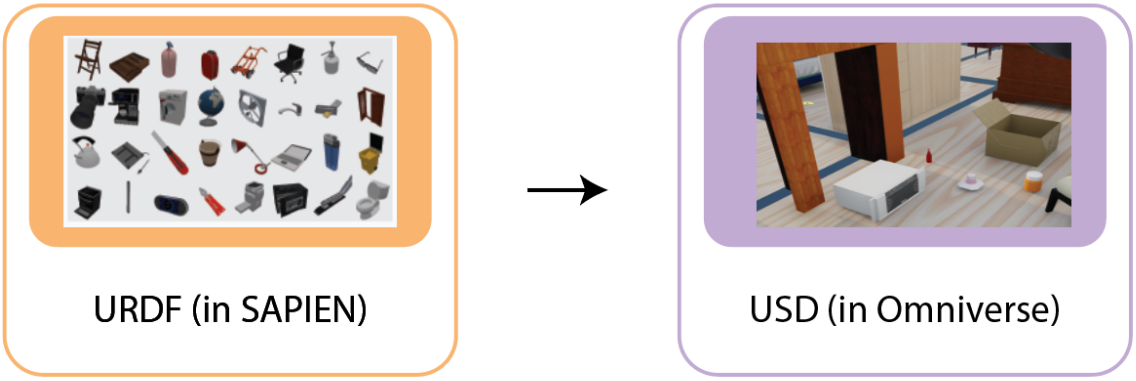
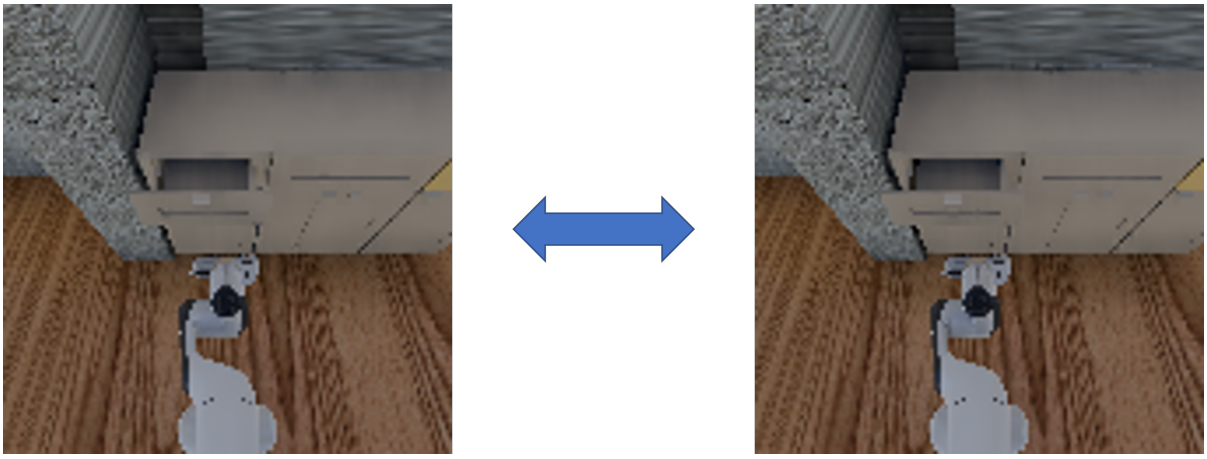


Figure A.2: Pipeline of parsing articulated bodies. To convert the original articulated bodies into USD format, we refine the built-in functionalities in Omniverse Isaac Sim and modify the assets manually.

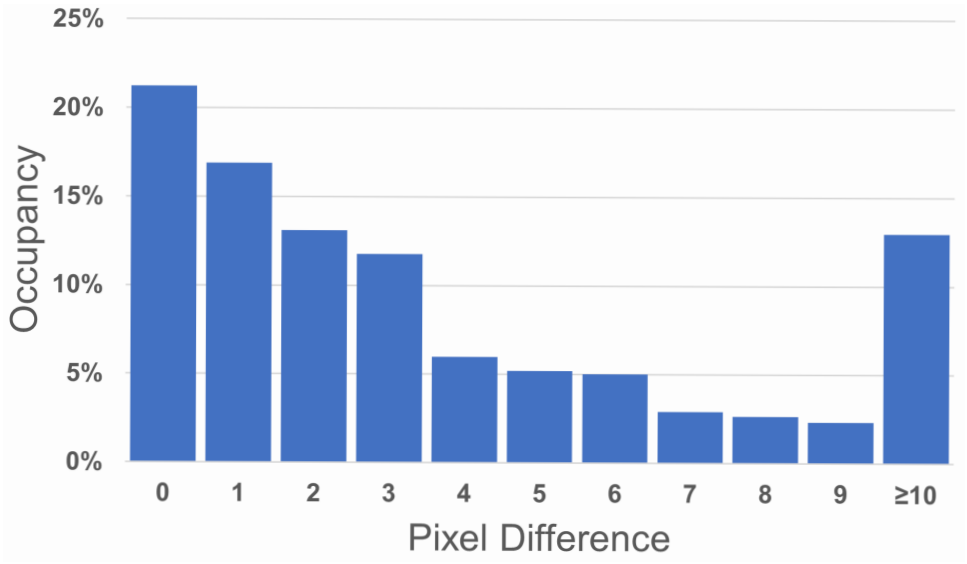
Working With Articulated Bodies: The articulated bodies in `ARNOLD` mainly come from SAPIEN. We use built-in tools of Omniverse Isaac Sim to parse the original articulated bodies (URDF) into USD format as shown in Figure A.2.

A.1.2 Speed

With five 128×128 cameras, our system runs at 17 fps for liquid simulation and 37 fps for rigid body simulations on an NVIDIA RTX 3090 GPU and AMD 5950X CPU. Currently, Omniverse Isaac Sim only supports serial cameras. We expect a huge performance improvement with the upcoming release of Omniverse Isacc Sim which supports parallel cameras.



(a) Two rendered images with a subtle difference.



(b) Distribution of pixel difference.

Figure A.3: An example that shows the rendering randomness in `ARNOLD`. (a) Two 128×128 images rendered from the same frame exhibit a subtle difference, with about 78.8% of the pixels being different. (b) The distribution of pixel difference. Here the difference per pixel is measured by the sum of RGB differences. More than 50% of the pixels differ less than a value of 3.

A.1.3 Randomness

Different from previous work (Zheng et al., 2022; Shridhar et al., 2022b), the rendering randomness in ARNOLD would result in non-deterministic images. This means two images rendered from the same frame may exhibit a subtle difference. Figure A.3 shows an example where about 78.8% of the pixels are different, together with the distribution of pixel difference.

A.2 Task Details

We provide illustrative examples of some tasks in Figure A.4. Figure A.5, Figure A.6, Figure A.7, and Figure A.8 illustrate the variations from different aspects in ARNOLD. In this section, we provide the implementation details of each task. In our notations, we use o to denote visual observation, a for action (*i.e.*, position and rotation with regard to the world). δ is a function to compute the pre-grasp pose.

A.2.1 PickupObject

Motion Planner: The motion planner of PickupObject consists of four sub-task stages. We depict each stage as follows, beginning with the visual observation and ending with a consequent end effector pose.

1. Observe o_1. Move the end effector to the pre-grasp pose. Reach $a_1 = \delta(a_2)$.
2. Observe o_2. Move the end effector to reach the position for grasping. Reach a_2.
3. Observe o_3. Close gripper. Reach a_3.
4. Observe o_4. Lift the object to the goal height. Reach a_4.

Learning and Evaluation: We extract two observation-action pairs for two-phase learning: o_1, a_2 and o_4, a_4. During evaluation, the robot executes the action predictions $\tilde{a}_2$ and $\tilde{a}_4$ similar to the motion planner:

1. Move to $\delta(\tilde{a}_2)$ for pre-grasp.

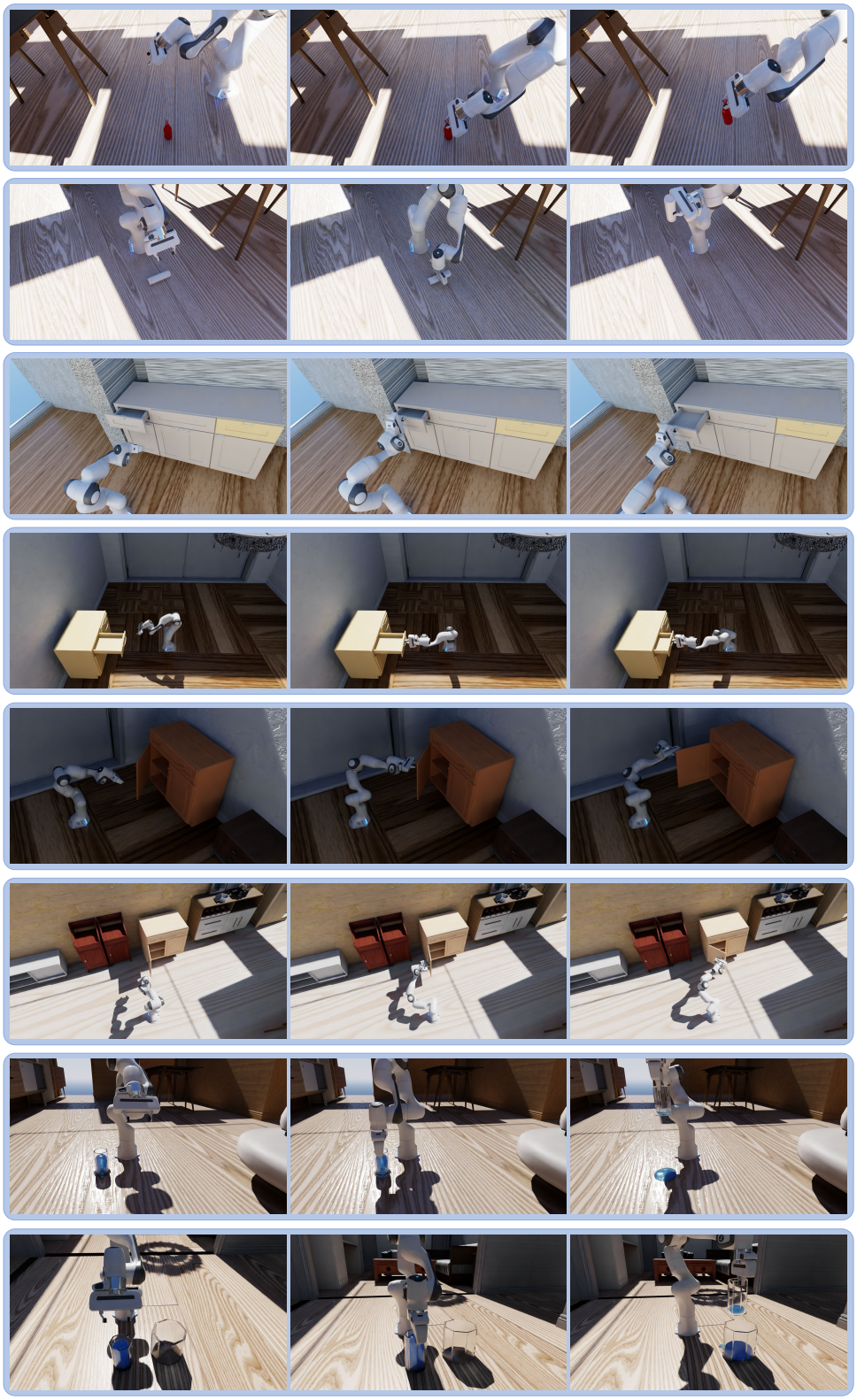


Figure A.4: Illustrations of the 8 tasks in ARNOLD

Figure A.5: Scene variations

Figure A.6: Object variations

Figure A.7: Lighting variations

Figure A.8: Material variations

2. Move to $\tilde{a}_2$ and close gripper.
3. Move to $\tilde{a}_4$ for goal state.

A.2.2 ReorientObject

Motion Planner: The motion planner of ReorientObject consists of four sub-task stages. We depict each stage as follows, beginning with the visual observation and ending with a consequent end effector pose.

1. Observe o_1. Move the end effector to the pre-grasp pose. Reach $a_1 = \delta(a_2)$.
2. Observe o_2. Move the end effector to reach the position for grasping. Reach a_2.
3. Observe o_3. Close gripper. Reach a_3.
4. Observe o_4. Reorient the object to goal orientation. Reach a_4.

Learning and Evaluation: We extract two observation-action pairs for two-phase learning: o_1, a_2 and o_4, a_4. During evaluation, the robot executes the action predictions $\tilde{a}_2$ and $\tilde{a}_4$ similar to the motion planner:

1. Move to $\delta(\tilde{a}_2)$ for pre-grasp.
2. Move to $\tilde{a}_2$ and close gripper.
3. Rotate to $\tilde{a}_4$ for goal state.

A.2.3 OpenDrawer and CloseDrawer

Motion Planner: The motion planner for these two tasks consists of four sub-task stages. We depict each stage as follows, beginning with the visual observation and ending with a consequent end effector pose.

1. Observe o_1. Move the end effector to the pre-grasp pose. Reach $a_1 = \delta(a_2)$.
2. Observe o_2. Move the end effector to reach the position for grasping. Reach a_2.
3. Observe o_3. Close gripper. Reach a_3.

4. Observe o_4. Gradually pull or push the drawer until the goal condition is satisfied. We apply linear interpolation on the action translation to slow down since excessive movement will result in the detachment of gripper. Reach a_4.

Learning and Evaluation: We extract two observation-action pairs for two-phase learning: o_1, a_2 and o_4, a_4. During evaluation, the robot executes the action predictions $\tilde{a}_2$ and $\tilde{a}_4$ similar to the motion planner:

1. Move to $\delta(\tilde{a}_2)$ for pre-grasp.
2. Move to $\tilde{a}_2$ and close gripper.
3. Gradually interpolate like the motion planner toward $\tilde{a}_4$ for goal state.

A.2.4 OpenCabinet and CloseCabinet

Motion Planner: The motion planner for these two tasks consists of four sub-task stages. We depict each stage as follows, beginning with the visual observation and ending with a consequent end effector pose.

1. Observe o_1. Move the end effector to the pre-grasp pose. Reach $a_1 = \delta(a_2)$.
2. Observe o_2. Move the end effector to reach the position for grasping. Reach a_2.
3. Observe o_3. Close gripper. Reach a_3.
4. Observe o_4. Gradually pull or push the cabinet until the goal condition is satisfied. We apply linear interpolation on the action translation to slow down, and apply spherical linear interpolation (Slerp) on the action rotation to meet the revolute constraint. Reach a_4.

Learning and Evaluation: We extract two observation-action pairs for two-phase learning: o_1, a_2 and o_4, a_4. During evaluation, the robot executes the action predictions $\tilde{a}_2$ and $\tilde{a}_4$ similar to the motion planner:

1. Move to $\delta(\tilde{a}_2)$ for pre-grasp.

2. Move to $\tilde{a}_2$ and close gripper.
3. Gradually interpolate like the motion planner toward $\tilde{a}_4$ for goal state.

A.2.5 PourWater and TransferWater

Motion Planner: The motion planner of these two tasks consists of seven sub-task stages. We explain each stage as follows, beginning with the visual observation and ending with a consequent end effector pose:

1. Observe o_1. Move the end effector to the pre-grasp pose. Reach $a_1 = \delta(a_2)$.
2. Observe o_2. Move the end effector to reach the position for grasping. Reach a_2.
3. Observe o_3. Close gripper. Reach a_3.
4. Observe o_4. Lift the cup up to the target height for pouring. Reach a_4.
5. Observe o_5. Translate the cup horizontally to the position for pouring. Reach a_5.
6. Observe o_6. Gradually tilt the cup to pour water out until the goal condition is satisfied. We apply spherical linear interpolation on the action rotation to slow down and make fluids controllable. Excessive tilting would suddenly empty the water. Reach a_6.
7. Observe o_7. Gradually rotate the cup back to an upright pose, similar to the previous stage. Reach a_7.

Learning and Evaluation: We extract two observation-action pairs for two-phase learning: o_1, a_2 and $o_4, a_{5,6}$, where $a_{5,6}$ combines the translation of a_5 and rotation of a_6. During evaluation, the robot executes the action predictions $\tilde{a}_2, \tilde{a}_{5,6}$ similar to the motion planner. Note that we maintain the angular velocity to a constant, which ensures reproducibility of the amount of poured water, as well as distinctions among the rotations for different goal states.

1. Move to $\delta(\tilde{a_2})$ for pre-grasp.
2. Move to $\tilde{a}_2$ and close gripper.
3. Lift up from $\tilde{a}_2$ to the height of $\tilde{a}_{5,6}$.

4. Translate horizontally to the position of $\tilde{a}_{5,6}$, with rotation unchanged.
5. Gradually tilt the cup to $\tilde{a}_{5,6}$ by spherical linear interpolation (Slerp).
6. Gradually rotate the cup back to the upright pose.

A.3 Data Collection

A.3.1 Human Annotations

We collect human annotations for the positions of robots and interactive assets to ensure reasonable configurations in 3D scenes. For the rationality of such annotations, we ask human operators to teleoperate robots to complete tasks. We will first introduce the settings of robot teleoperation in Section A.3.1.1 and then present the whole pipeline of data collection in Section A.3.1.2.

A.3.1.1 Robot Teleoperation

Frame Definition: Human operators use an Xbox controller to control the robot and the camera for data collection. The input to the controller is in the robot base frame, where the X axis originates from the robot's base and points to the object, and the Y axis is the upward pointing axis, as displayed in Figure A.9. This controller input is transformed into the world frame for robot motion at each timestep.

Robot Control: Human operators can adjust the position and rotation of the robot end effector, as well as toggle the gripper. The Xbox layout is shown in Figure A.10. Specifically, the controller supports two control modes for rotation: joint position control and end effector rotation control. The joint position control mode allows the operator to directly change the positions of joints: A1 (shoulder joint), A6 (forearm joint), and A7 (wrist joint). The end effector rotation control mode allows the operator to rotate the end effector around the X, Y, Z axis of the robot base frame while maintaining its position, which is more general for rotation control. Operators can switch between these

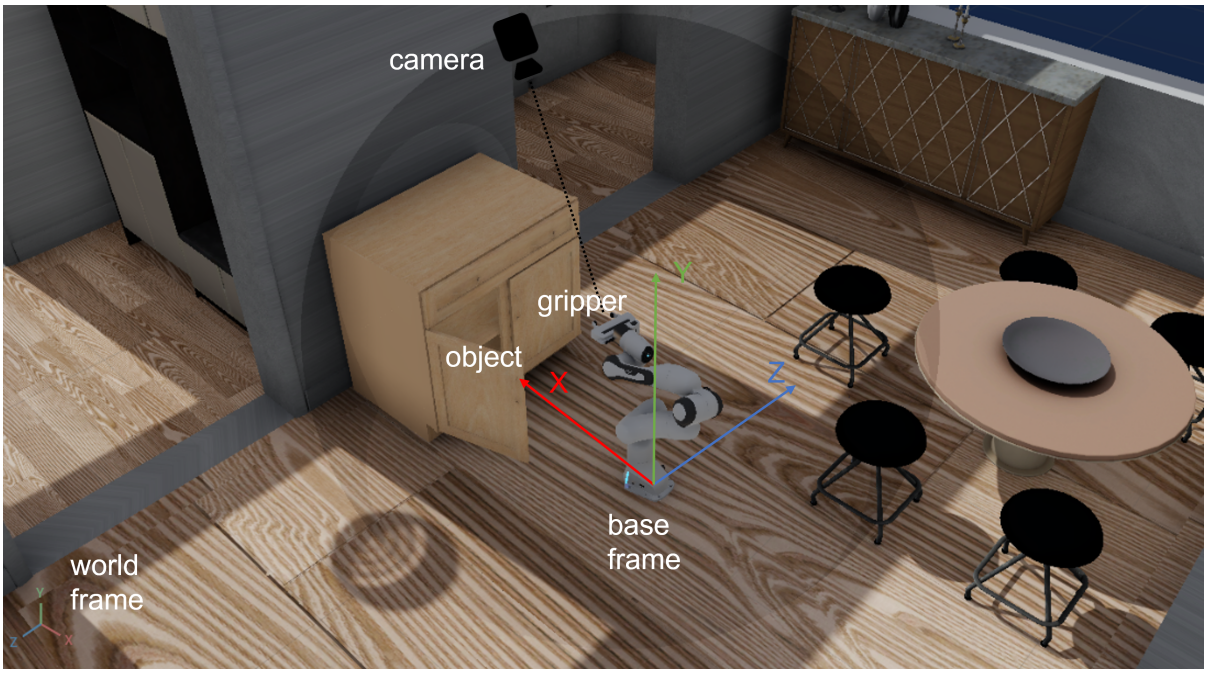


Figure A.9: An illustration of the frame and camera for robot teleoperation

two modes during the data collection process.

Camera Control: Human operators can move their viewing camera freely in the 3D scenes to avoid occlusion. This is accomplished by a spherical camera control (Figure A.9), where the camera can move on a sphere centered at the robot end effector while keeping casting toward the end effector. Note that the radius of the sphere can also be adjusted for a clearer view.

A.3.1.2 Collection Pipeline

Settings: The collection of human annotations is conducted through a user interface (UI), which is implemented as an extension of Omniverse Isaac Sim, as shown in Figure A.11. For each task, we enumerate the compositions of objects, scenes, initial states and goal states. Each composition instance is called a mission. Human annotators are supposed to annotate each mission with two configurations of the relative positions between the robot and object. Each configuration is loaded for human control for two trials. Hence, each mission can produce up to four trajectories.

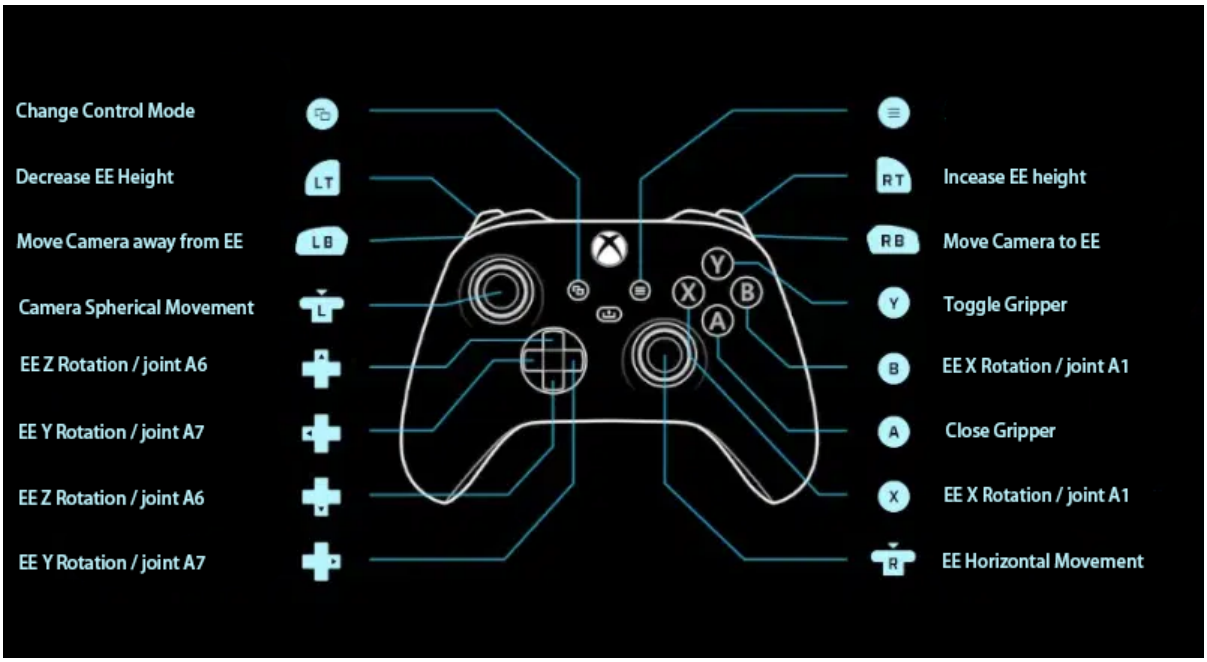


Figure A.10: A schematic of the Xbox controller

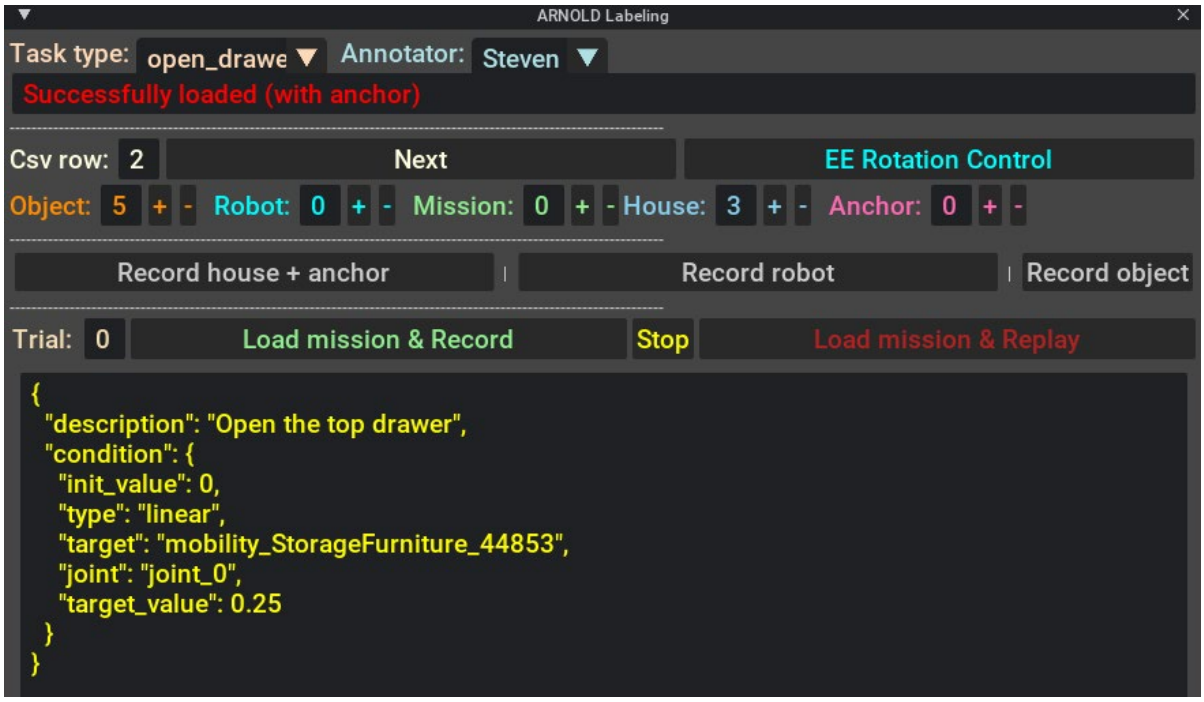


Figure A.11: A toy example of the user interface (UI) for collecting human annotations

Procedures: The annotator starts annotating a mission by placing the object group (robot and object) at an appropriate anchor position in the scene and clicking the "Record house + anchor" button. Then, the annotator is supposed to adjust the relative position between the robot and object before clicking the "Record robot" and "Record object" buttons. Clicking these two buttons amounts to recording a configuration. For each configuration, the annotator is supposed to control the robot to complete the task after clicking the "Load mission & Record" button. Each click is regarded as a trial. Once the annotator succeeds, a message of "Task Success" will be displayed. Then the annotator can click the "Stop" button and thereby produce a human trajectory. The trajectory replay is supported by clicking the "Load mission & Replay" button. Altogether, each mission is annotated with two configurations and two trials for each configuration. To proceed to the next mission, the annotator can click the "Next" button.

Statistics: We record the human trajectories with the aforementioned Xbox controller at 120Hz and finally collect 2990 trajectories. These trajectories consist of 6.2M frames (14.3 hours) in total and 2073 frames (17.3 seconds) on average. The minimum, median and maximum length are 110 frames (0.9 seconds), 1752 frames (14.6 seconds) and 11336 frames (94.5 seconds), respectively. The distribution is shown in Figure A.12. Although these human trajectories are not directly used for training in `ARNOLD`, the annotations of configurations and keypoints are referenced by motion planners for demonstration generation. And these human trajectories are of potential use when more powerful algorithms are available.

A.3.2 Data Augmentation

For richer data variations, we apply augmentation to the robot positions based on collected human annotations. For example, the robot positions may be expanded by shifting 10cm horizontally along four orthogonal directions. After that, we use the motion planner to check if the robot can execute the task successfully with new positions.

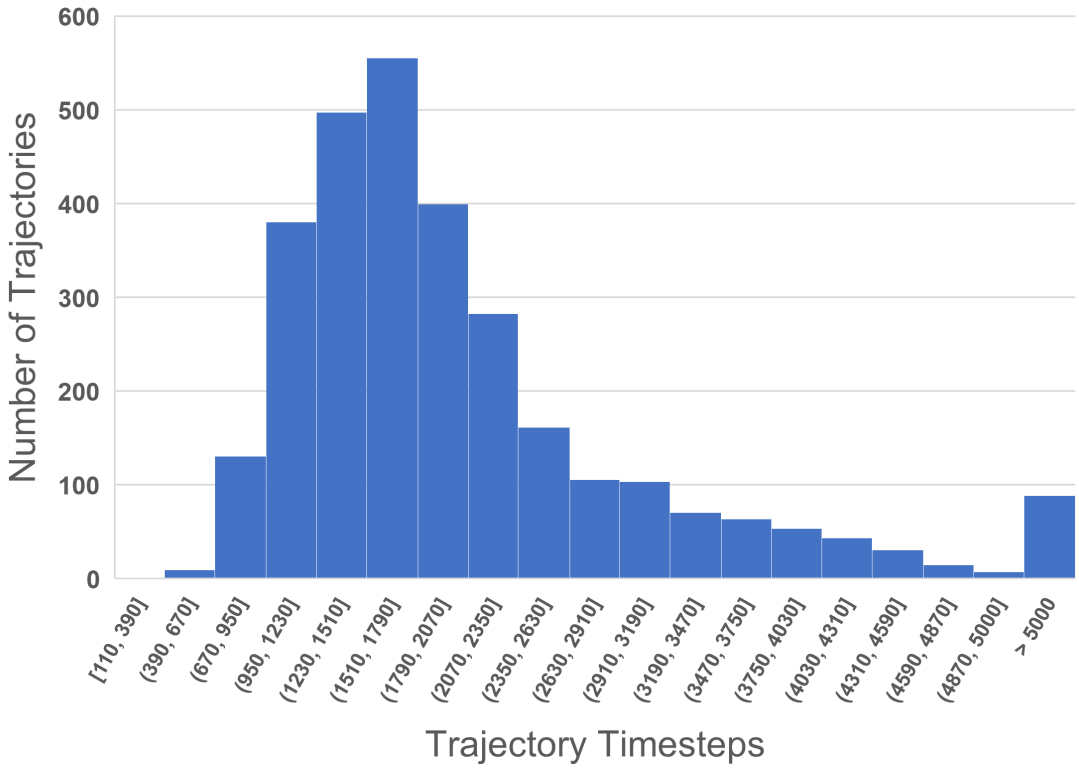


Figure A.12: Distribution of human-annotated trajectory length (timesteps). Here 120 timesteps amount to a second.

Tasks	Examples of Delexicalized Templates
PickupObject	*Raise* [value_object] [value_height] *above the ground*
ReorientObject	*Reorient* [value_object] [value_degree] *away from the up axis*
OpenDrawer	*Open the* [value_position] [value_object] [value_percent]
CloseDrawer	*Close the* [value_position] [value_object] [value_percent]
OpenCabinet	*Open the* [value_position] [value_object] [value_percent]
CloseCabinet	*Close the* [value_position] [value_object] [value_percent]
PourWater	*Pour* [value_percent] *water out of* [value_object]
TransferWater	*Transfer* [value_percent] *water to* [value_object]

Table A.1: A few examples of delexicalized instruction templates for various tasks. During data generation, we first sample a delexicalized instruction template from the template pool. Then, we fill in the placeholders by sampling from candidate pools that consist of equivalent phrases.

A.3.3 Language Instructions

We sample a few instruction templates from our template pool and present them in Table A.1. For generality and diversity, we construct these templates with placeholders, which can be lexicalized flexibly. In these templates, "[value_object]" holds for the actual object name, *e.g.*, "*white bottle*". The goal states are specified by "[value_height]" (*e.g.*, "20*cm*"), "[value_degree]" (*e.g.*, "90 *degrees*"), and "[value_percent]" (*e.g.*, "40%"). In some tasks where multiple objects exist, the referential words are necessary and specified by "[value_position]", which indicates the positional information of the target object, *e.g.*, "*top left*". The placeholders are lexicalized randomly by sampling from candidate pools which contain equivalent phrases, *e.g.*, "*fifty percent*", "*half*", "*two quarters*".

A.3.4 Verification

We replay the recorded keypoints of the generated demonstrations in the procedure of evaluation to verify their validity. After removing the failed demonstrations, we obtain a set of 10k demonstrations whose success is ensured achievable.

A.4 Implementation Details

A.4.1 Additional Evaluation of BC-Z

BC-Z (Jang et al., 2021) is a behavior-cloning model that pursues zero-shot task generalization with large-scale data. We follow the implementation of PerAct (Shridhar et al., 2022b) to adapt the language-conditioned model (BC-Lang) to our settings. This model takes as input a single-view RGB-D image, with RGB and depth processed by two separate streams. We select the front view for the single-view visual input. The task instruction is processed by a CLIP text encoder (Radford et al., 2021) to extract a global semantic embedding. With the visual and language input, BC-Lang outputs directly regresses an end effector pose, whose translation and rotation are both continuous values (coordinates and quaternions). There are two variants of backbone: CNN and ViT. We

	Test		Object		Scene		State		Any State	
BC-Lang-CNN	2.52	14.29	0.00	0.65	1.18	5.88	0.00	1.44	0.84	0.84
BC-Lang-ViT	5.88	19.33	0.00	0.00	0.39	2.35	0.86	4.02	0.84	5.04
PerAct	31.09	44.54	6.45	21.29	20.78	31.37	10.92	12.64	21.85	30.25

Table A.2: The performances of two BC-Lang variants. We add the performance of PerAct for comparison. The results show that BC-Z, regressing end effector pose with a single-view image as visual input, is much less effective.

run additional experiments with the two BC-Lang variants and report their performances on the `OpenDrawer` task (shown in Table A.2). The results indicate BC-Z cannot handle the tasks in `ARNOLD`.

A.4.2 Workflow

Data Sampling: We consider three aspects of balance when sampling data for training. (1) To ensure object-level balance, we categorize the demonstrations according to the manipulated object. A uniform sampling over categories is prior to sampling a demonstration within the category. (2) To ensure phase-level balance, we use a biased weight of 1:4 to sample a phase for training based on a selected demonstration. (3) To ensure task-level balance during multi-task training, a uniform sampling over tasks is prior to the task-specific sampling.

Validation: We perform model selection by evaluating each checkpoint instead of tracking the loss on *Val* set since we observe an inconsistency between these two metrics, as mentioned by Shridhar et al. (2022a). We select the best checkpoint for final evaluation.

A.4.3 Ablation

Without Language: We simply skip the operation of appending language embedding to the flattened voxel grid, making the latent representation contain only visual information.

State Head: As a naive implementation, we represent all object states through a normalized value in the $[0, 1]$ range. For PickupObject, the state value is expressed by

$\frac{h}{40}$ (initially 0), where h is the height in cm. For ReorientObject, the state value is expressed by $\frac{\alpha}{180}$ (initially 0.5), where α is the angle (degrees) between object orientation and the upward axis. For the rest of the tasks, the state value is equal to the percentage. The state head predicts both the current state and the goal state, which are optimized via MSE loss. Although learning to regress state values brings moderate improvements, there is still quite a large space for better methods on state modeling.

Language Encoder: We adopt T5-base encoder (Raffel et al., 2020) as the alternative to CLIP language encoder. To ensure a fair comparison, we pad the token sequence encoded by T5-base to 77, the same as the token sequence of CLIP. Despite the sophisticated pre-training on large-scale text corpus, T5-base as a language encoder for language-grounded task may be limited by a lack of alignment with vision modality. Moreover, the scarce data in robotics tasks may induce T5-base to suboptimal performances.

A.4.4 Sim2Real

Equipment and Configurations: To set up the real-robot experiment, we adopted the Franka Emika Panda robot arm, the RealSense D435 RGB-D camera, and a random common object in the object category. We placed the robot arm and object similar to the configurations in simulation. Notably, we only used the left camera view, and nothing was tuned particularly. We used the aruco marker to calibrate the D435 camera and Franka API to initialize the robot arm.

Inference: We pre-processed the point cloud due to the imperfect depth camera, *e.g.*, discarding noisy or too far away points. Next, we fetched the PerAct model trained in the simulator to perform inference for the next movement of the robot arm. We utilized Franka API to execute the actions and evaluate similarly to the metrics in `ARNOLD`.

Results: We summarize the real-world experiment as follows. (1) We experimented with opening/closing two different drawers and picking up five objects. We conducted 19,

8, and 31 trials in various configurations (*e.g.*, object poses, camera poses, instructions *etc.*) for closing drawers, opening drawers, and picking up objects, respectively. And the corresponding numbers of success are 6, 0, and 4. (2) Throughout our experiments, we observe preliminary Sim2Real transfer capabilities, *i.e.*, reasonable predictions for picking up objects and manipulating drawers. Nonetheless, the complex real-world environments, *e.g.*, strict friction and sensory noise, still limit the performance of Sim2Real transfer to a considerable extent.

APPENDIX B

MindAgent Details

B.1 Prompt Examples

We provide some examples of prompts for CuisineWorld. Figure B.1 shows an example of the system prompt info. Figure B.2 shows an example of a partial demonstration.

```
The available actions are :
1) goto: goto a tool location
2) get: get some object from a tool
3) put: put some object into a tool
4) activate: activate the tool to cook all ingredients inside the tool into a different tools
5) noop: not performing any actions
Sometimes the system will give you error messages. Please consider these error messages when executing actions.
You need to specify action for all of the agents, **except human**. They all have different agent numbers. Do not assign actions to the same agent more than once.

When the tools reach its capacity, you need to take stuff out. Otherwise, you cannot put items inside.
When you are holding objects, you cannot get any more objects.
When you are holding objects, you cannot activate tools.
After you cooked a required dish, you need to put it into the servingtable.
You can only pick up objects from the tool location, if you are located at the tool location.
When you activate any tools, make sure all the items inside the tool are respecting the recipes. Otherwise, you will cook waste. Avoid waste at all cost.
*** You should mix salad in the mixer. To make salad you should chop veggies first. ***
*** If the tool is occupied, indicated by the occupy() predicate, you cannot get objects from it or put objects into it. ***
*** The food orders are keep coming. You should finish as many dishes as possible and finish every dish as soon as possible. Please deliver the order to the serveringtable when it is finished. ***
*** The dish will expire after the lifetime reaches 0 and it's not at the serveringtable. Please avoid this. *** Here are the recipes:

Cook porkMeatcake at:
 -- location: blender
 -- with ingredients:     pork,     flour,
Cook salmonSashimi at:
 -- location: chopboard
 -- with ingredients:     salmon,
Cook tunaSashimi at:
 -- location: chopboard
 -- with ingredients:     tuna,
Cook mixedSashimi at:
 -- location: mixer
 -- with ingredients:     salmonSashimi,     tunaSashimi,
The following objects are available:
 --1) salmonSashimi
 --2) tuna
 --3) mixedSashimi
 --4) tunaSashimi
 --5) porkMeatcake
 --6) salmon
 --7) flour
 --8) pork
The objecsts are cooked using tools or are just base ingredients.
Among them, the following are base ingredients:
 --1) tuna
 --2) salmon
 --3) flour
 --4) pork
You can only obtain base ingredients from the storage initially.
Additional rules:
You can place up to infinite item into the storage0
You can place up to infinite item into the storage0
You can place up to infinite item into the servingtable0
You can place up to infinite item into the servingtable0
You can place up to 1 item into the chopboard0
You can place up to 1 item into the chopboard0
You can place up to 1 item into the chopboard1
You can place up to 1 item into the chopboard1
You can place up to 5 item into the mixer0
You can place up to 5 item into the mixer0
You can place up to 5 item into the mixer1
You can place up to 5 item into the mixer1
** Only ** the following tools are available:
storage0, servingtable0, chopboard0, chopboard1, mixer0, mixer1, You cannot pick up these tools. You can only use those tools at the corresponding location.
```

Figure B.1: The MindAgent system prompt example

```
There are 2 agents available, so you can execute 2 actions at a time.
Goal: porkMeatcake
t=0
-state:
at(agent0, servingtable0)
at(agent1, servingtable0)
hold(agent0, None)
hold(agent1, None)
inside(storage0, None)
inside(blender0, None)
inside(chopboard0, None)
inside(servingtable0, None)

-action:
***
goto_agent0_storage0
goto_agent1_storage0
***

Goal: porkMeatcake
t=1
-state:
at(agent0, storage0)
at(agent1, storage0)
hold(agent0, None)
hold(agent1, None)
inside(storage0, None)
inside(blender0, None)
inside(chopboard0, None)
inside(servingtable0, None)

-action:
***
get_agent0_flour_storage0
get_agent1_pork_storage0
***

Goal: porkMeatcake
t=2
-state:
at(agent0, storage0)
at(agent1, storage0)
hold(agent0, flour)
hold(agent1, pork)
inside(storage0, None)
inside(blender0, None)
inside(chopboard0, None)
inside(chopboard1, None)
inside(pan0, None)
inside(servingtable0, None)

-action:
***
goto_agent0_blender0
goto_agent1_blender0
***
```

Figure B.2: The `MindAgent` system partial one-shot demo example

B.2 Prompt Engineering Details

Our initial approach began with a simple and generic prompt, designed to be as neutral as possible. This baseline prompt was tested across all models, including GPT-4, Claude, and other LFMs in our study. We observed that only GPT-4 and Claude were able to generate reasonable planning responses with this initial prompt. Consequently, we refined our prompt engineering efforts to better suit these two models, aiming to optimize their performance and response quality.

For the other LFMs, we also attempted individual prompt engineering. However, these efforts did not yield significant improvements in their responses compared to the initial

trial. After several rounds of testing and refinement, we concluded that further prompt customization for these models did not result in notably better outcomes.

As a result, we decided to use a consistent prompt across all LFMs for the final comparison. This decision was made to maintain uniformity in testing conditions, despite the prompts being more closely tuned to GPT-4 and Claude. We believe this approach offers a reasonable balance between fairness and practicality in evaluating the capabilities of different LFMs under similar conditions.

B.3 LFM Settings

We perform experiments on `CuisineWorld` through OpenAI APIs and Anthropic APIs (Anthropic, 2023). All GPT-4 (OpenAI, 2023) experiments employ the gpt-4-0613 model, and all Chat-GPT (Ouyang et al., 2022) experiments employ gpt-3.5-turbo-0613. For the Llama 2 (Touvron et al., 2023) experiments, we use the hugging face inference endpoints Llama-2-70b-chat-hf. We set the temperature for all experiments to 0.1 following Wang et al. (2023a). We report the average results over three episodes. These LFMs can be accessed through publicly available APIs or huggingface endpoints. These APIs are for public use. However, users should be aware of their licensing agreements before using them.

B.3.1 Design Considerations

We adopted a centralized setting for the following reasons:

Token Limitation: In decentralized settings like ours, each agent needs to compile a full system message which includes a distinct copy of recipes, rules, one-shot demo is required, while in centralized setting we only need one system message to describe them, which significantly reduces the total number of input tokens (within the input context) of LFM and therefore make the framework more affordable.

Communication Overhead: Each agent needs to receive both their own state and the states of the other agents, possibly in text, while in centralized setting there is no communication among agents as other states are directly observable. This can also save the cost on input tokens.

Save API Call: In decentralized setting, to generate one environment step, we need to call API N times (N is the number of agents), while in centralized setting we only need to call the API once and we will be able to obtain actions for all agents (as produced by the centralized LFM dispatcher).

B.4 `CuisineWorld` Task Details

B.4.1 `CuisineWorld` Task Definitions

We follow prior work (Yao et al., 2023; Liu et al., 2023; Deng et al., 2023) to **interactively evaluate LFMs as planning agents**. Overall, the interactive evaluation can be formulated as a *Markov Decision Process* $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{G})$, with state space $\mathcal{S}$, action space $\mathcal{A}$ (effectively indicating all the possible schedules that can be made at a single time step), transition dynamics $\mathcal{T}$, reward function $\mathcal{R}$, and task instruction space $\mathcal{G}$. Note that, although there are multiple agents inside `CuisineWorld` that can be coordinated, asmentioned above, we adopt a centralized planning scheme and thereby formulate our game as a single-agent, fully-observable decision-making problem. Figure 1.2 illustrates the state and action space as well as the possible tasks of our game.

State Space $\mathcal{S}$: In a `CuisineWorld` virtual kitchen, there are two types of entities: `location` and `agent`. For each entity, the game will provide a set of descriptions, and the aggregated descriptions of all entities will be the state returned by the game. A `location` can be *storage*, where one can obtain ingredients and dispense waste, a *serving table*, onto which one should put the completed, or a cooking tool; *e.g.*, *pan* or *blender*. We offer up to two descriptions for each location: `inside(location, items)`, indicating

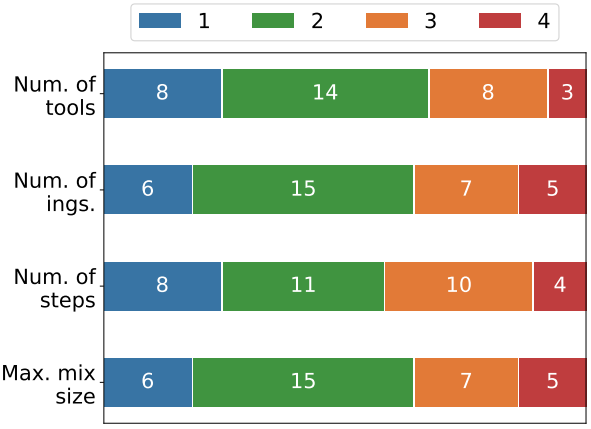


Figure B.3: Dish distribution over the number of tools and ingredients (ings.) involved, cooking steps, and maximum mixture size as in the recipe.

what items (some ingredients, completed dishes, *etc.*) are now inside the location, and `occupy(location)`, suggesting `location` is now being used and cannot be touched; *e.g.*, an activated blender. An `agent` is an entity that can be dispatched to complete the task, and we provide up to three descriptions for each agent: `at(location, agent)`, indicating that `agent` is now at `location`, `hold(agent, items)`, suggesting what items `agent` is holding, and `occupy(agent)`, implying `agent` is now operating a tool, *e.g.*, chopping some fruits, and will not respond to any dispatching command. The set of tool distributions can be found in Figure B.3.

Action Space $\mathcal{A}$: An action in `CuisineWorld` is a list of dispatching commands. Given N `agent` entities, a total of N commands must be generated. The agent provides the following commands (also tabulated in Table B.1):

1. `goto(agent, location)`, to let `agent` move to `location`;
2. `get(agent, location, item)`, to let `agent` get a specific item from `location`;
3. `put(agent, location)`, to put whatever `agent` is holding into `location`;
4. `activate(agent, location)`, to let `agent` turn on `location` if it is a cooking tool, *e.g.blender*;

Type	Arguments	Description
`goto`	`agent location`	Move `agent` to `location`
`get`	`agent location (item)`	`agent` obtain `item` from `location`
`put`	`agent location`	`agent` put everything it holds to `location`
`activate`	`agent location`	`agent` turn on `location`
`noop`	`agent`	not dispatching `agent`

Table B.1: Action space in `CuisineWorld`.

5. `noop(agent)`, to have `agent` perform no actions in this round of dispatching.

Note that, to avoid the possible confusion of multiple agents being dispatched to operate with the same `location`, the dispatcher also must properly order the dispatching commands as they will be executed sequentially.

B.4.2 Implementing `CuisineWorld`

The implementation of `CuisineWorld` mostly follows the spirit of *Overcooked!*, a renowned video game. Therefore, we refer to many of its game mechanisms while simplifying some of them; *e.g.*, we skip low-level control and assume all `agent` entities have access to all `location` at any time. Specifically, we crawled the rules and recipes from the community-contributed wiki[4] of *Overcooked!*, streamlined them, and made necessary modifications, ending up with the basic version of `CuisineWorld` comprising **10** types of `location` (*serving table*, *storage*, and 8 different cooking tools), **27** types of ingredients, and **33** unique dishes. We grouped the dishes based on their difficulty (primarily based on the number of cooking tools involved) to design and implement **12** game levels, which are further categorized into 4 classes: *entry*, *simple*, *intermediate*, and *advanced*, with 3 levels

[4]Steam community wiki

2-agent	very simple			simple			intermediate			advanced			Avg.
	level 0	level 1	level 7	level 2	level 4	level 8	level 3	level 9	level 10	level 5	level 11	level 12	
GPT4 $\tau_{\text{int},(1)}$	18/54	18/56	12/31	14/34	12/30	3/30	10/26	7/20	7/23	6/23	6/21	10/36	0.318
GPT4 $\tau_{\text{int},(2)}$	18/31	17/34	10/23	13/26	12/22	9/22	10/17	8/11	6/12	5/13	4/14	8/21	0.486
GPT4 $\tau_{\text{int},(3)}$	18/25	19/25	10/17	16/18	11/18	6/16	11/13	6/8	7/10	8/10	9/9	8/17	0.709
GPT4 $\tau_{\text{int},(4)}$	18/18	18/19	12/12	11/14	11/12	7/11	12/12	8/8	9/9	6/7	8/9	11/12	**0.912**
GPT4 $\tau_{\text{int},(5)}$	18/18	17/17	12/12	11/13	11/13	9/9	11/11	4/5	7/7	8/8	8/8	9/12	**0.937**
CoS	0.727	0.706	0.682	**0.687**	**0.664**	0.504	0.764	0.725	0.701	0.661	0.692	0.559	0.673

Table B.2: 2 agents performance on different tasks

3-agent	very simple			simple			intermediate			advanced			Average
	level 0	level 1	level 7	level 2	level 4	level 8	level 3	level 9	level 10	level 5	level 11	level 12	
GPT4 $\tau_{\text{int},(1)}$	21/55	24/55	16/33	17/33	9/28	6/32	12/25	5/20	8/21	7/22	7/22	9/26	**0.368**
GPT4 $\tau_{\text{int},(2)}$	20/31	25/33	11/22	4/24	13/24	7/21	14/20	9/12	9/13	7/14	8/14	10/23	0.549
GPT4 $\tau_{\text{int},(3)}$	22/25	21/26	17/17	11/20	9/17	4/15	13/14	8/8	12/12	7/7	9/10	10/16	**0.791**
GPT4 $\tau_{\text{int},(4)}$	22/22	20/21	14/14	9/13	7/10	6/10	10/10	6/7	10/10	5/8	7/8	11/13	0.846
GPT4 $\tau_{\text{int},(5)}$	20/20	15/16	11/12	10/14	10/11	8/9	12/12	6/6	8/8	5/5	8/8	6/10	0.914
CoS	**0.781**	**0.778**	**0.780**	0.528	0.600	0.455	0.822	**0.771**	**0.815**	0.689	**0.733**	**0.570**	**0.694**

Table B.3: 3 agents performance on different tasks

4-agent	very simple			simple			intermediate			advanced			Average
	level 0	level 1	level 7	level 2	level 4	level 8	level 3	level 9	level 10	level 5	level 11	level 12	
GPT4 $\tau_{\text{int},(1)}$	22/54	18/55	17/34	13/34	8/28	9/33	16/27	5/20	8/23	5/22	8/22	8/35	0.349
GPT4 $\tau_{\text{int},(2)}$	24/32	21/33	14/24	14/25	12/24	11/22	16/19	7/12	9/15	7/14	6/12	12/23	**0.590**
GPT4 $\tau_{\text{int},(3)}$	23/25	23/26	13/18	11/19	10/17	11/17	15/17	8/9	11/11	7/8	10/11	9/17	0.785
GPT4 $\tau_{\text{int},(4)}$	22/22	21/22	14/14	7/15	10/13	10/12	12/13	9/9	10/10	6/7	8/8	9/13	0.875
GPT4 $\tau_{\text{int},(5)}$	14/18	20/20	14/14	7/13	9/11	7/8	12/12	5/5	7/7	6/6	3/5	7/10	0.859
CoS	0.771	0.761	0.761	0.505	0.592	**0.626**	**0.848**	0.744	0.790	**0.692**	0.675	0.534	0.692

Table B.4: 4 agents performance on different tasks

each. Note that the recipes, dishes, and levels can be easily extended to incorporate more challenging tasks.

B.4.3 Task Graph Visualization

In `CuisineWorld`, we provide tasks of different complexities to holistically evaluate the multi-agent system's performance. Additionally, the environment is highly customizable and extendable. Users only need only modify the JSON files to add more tasks or modify existing tasks. We visualize different `CuisineWorld` task graphs in Figure B.4 through Figure B.16. The dish distribution is shown in Figure B.3.

	2 agent				3 agent				4 agent			
	GPT-4	Claude-2	LLaMA	ChatGPT	GPT-4	Claude-2	LLaMA	ChatGPT	GPT-4	Claude-2	LLaMA	ChatGPT
$\tau_{\text{int},(1)}$	10/26	3/24	0	0/24	12/25	5/26	0	0/24	16/27	9/25	0	0/24
$\tau_{\text{int},(2)}$	10/17	3/16	0	0/15	14/20	4/16	0	0/15	16/19	4/15	0	0/15
$\tau_{\text{int},(3)}$	11/18	3/12	0	0/12	13/14	3/12	0	0/12	15/17	4/12	0	0/12
$\tau_{\text{int},(4)}$	11/13	3/9	0	0/9	10/10	5/11	0	0/9	12/13	6/11	0	0/9
$\tau_{\text{int},(5)}$	11/11	4/6	0	0/6	12/12	5/7	0	0/6	12/12	6/7	0	0/6
CoS	0.686	0.3125	0	0	0.822	0.372	0	0	0.848	0.473	0	0

Table B.5: Performance of other LFMs on Level 3

2 agent	GPT-4	GPT-4 w/ few-step	GPT-4 w/o inference knowledge	GPT-4 w/o feedback
$\tau_{\text{int},(1)}$	10/26	8/26	8/25	4/25
$\tau_{\text{int},(2)}$	10/17	11/19	9/17	4/17
$\tau_{\text{int},(3)}$	11/13	11/13	10/12	4/12
$\tau_{\text{int},(4)}$	12/12	9/11	8/9	1/9
$\tau_{\text{int},(5)}$	11/11	10/10	9/9	5/7
CoS	0.764	0.710	0.714	0.311

Table B.6: Additional ablation results

level_3	4agent using 4agent demo	4agent using 2agent demo	3agent using 3agent demo	3agent using 2agent demo
GPT4 $\tau_{\text{int},(1)}$	16/27	14/27	12/25	11/25
GPT4 $\tau_{\text{int},(2)}$	16/19	16/20	14/20	11/19
GPT4 $\tau_{\text{int},(3)}$	15/17	15/16	13/14	12/14
GPT4 $\tau_{\text{int},(4)}$	12/13	13/13	10/10	12/12
GPT4 $\tau_{\text{int},(5)}$	12/12	12/12	12/12	11/11
CoS	0.848	0.851	0.822	0.775

Table B.7: Using different numbers of agents demos

B.5 Additional Results in `CuisineWorld`

Table B.2 through Table B.7 report additional performance results for several different numbers of agents and task complexity levels, performance of other LFMs, and additional ablation results. Table B.8 shows the results of using a centralized PPO agent comparing against our method using heavily engineered dense rewards.

B.5.1 Comparison Between `CuisineWorld` and Related Benchmarks

Table B.9 compares `CuisineWorld` against related benchmarks along the following criteria:

- **Multi-task**: The benchmark contains multiple different tasks.
- **Object Interaction**: Agents must manipulate or engage with different items or environmental elements to achieve certain goals with irreversible actions.

model	level 1	level 4
RL Performance	0.451	0.598
Ours(GPT4 $\tau_{int(4)}$)	0.947	0.917

Table B.8: Performance of masked PPO with 2 agents in level 1 and level 4.

Benchmark	Multi-task	Object Interaction	Tool Use	Maximum Agents	Collaboration	Human in-the-loop	Procedural Level Generation
ALFWorld (Shridhar et al., 2020b)	✓	✓	✓	1	✗	✗	✗
WAH (Puig et al., 2020)	✓	✓	✗	2	✓	✓	✗
TextWorld (Côté et al., 2019)	✓	✓	✓	1	✗	✗	✓
Generative Agents (Park et al., 2023)	✓	✓	✓	25	✗	✗	✓
EMATP (Liu et al., 2022c)	✓	✓	✓	2	✓	✗	✗
Overcooked-AI (Carroll et al., 2019)	✗	✓	✓	2	✓	✓	✗
HandMeThat (Wan et al., 2022)	✓	✓	✓	2	✓	✗	✗
DialFRED (Gao et al., 2022)	✓	✓	✓	2	✓*	✗	✗
TEACH (Padmakumar et al., 2022)	✓	✓	✓	2	✓*	✗	✗
CerealBar (Suhr et al., 2019)	✗	✗	✗	2	✓	✗	✗
LIGHT (Urbanek et al., 2019)	✓	✗	✗	1369	✗	✓	✓
Diplomacy (Bakhtin et al., 2022)	✗	✗	✗	7	✓	✓	✗
CordialSync (Jain et al., 2020)	✗	✓	✗	2	✓	✗	✗
CoELA (Zhang et al., 2023b)	✓	✓	✗	2	✓	✓	✗
TooManyCooks (Wu et al., 2021)	✓	✓	✓	2	✓	✓	✗
`CuisineWorld` (Ours)	✓	✓	✓	4+	✓	✓	✓

Table B.9: Comparison between `CuisineWorld` and other related benchmarks. *: Notably, even though multiple agents can be present, the second agent is limited to communicating with the first agent. The second agent cannot interact with the environment in an active gaming capacity.

- **Tool Use**: Completing tasks necessitates the use of specific tools by the agents.
- **Maximum Agents**: Denotes the upper limit of agents that can be present in any experiment.
- **Collaboration**: Many tasks mandate teamwork and collaboration between different agents.
- **Human in-the-loop**: The framework allows humans to join the game and collaborate actively with the agents.
- **Procedural Level Generation**: There is flexibility in adding new tasks, making the game dynamic and adaptable.

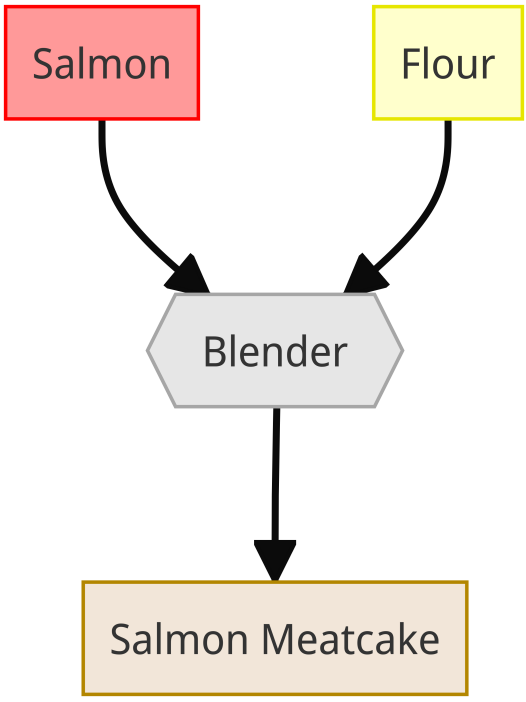
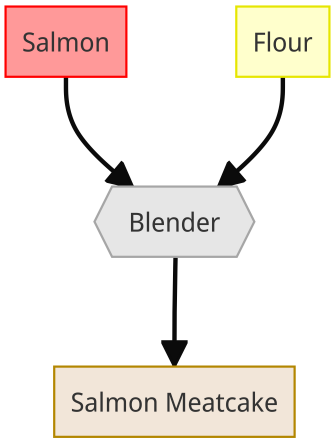
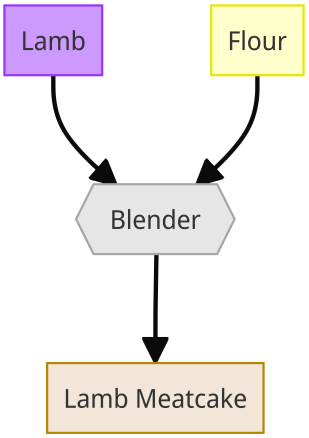


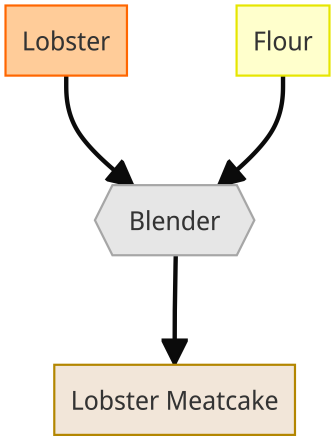
Figure B.4: Level 0 — Very Simple Salmon Meatcake



(a) Salmon Meatcake

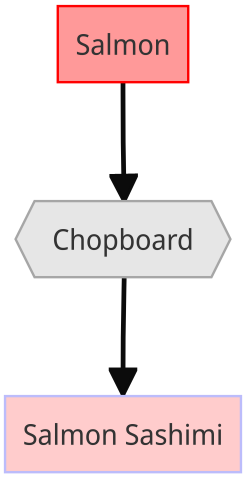


(b) Lamb Meatcake

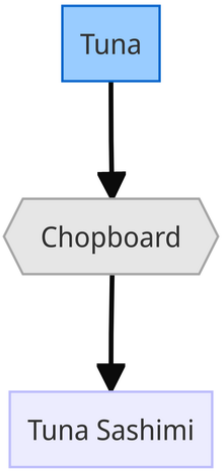


(c) Lobster Meatcake

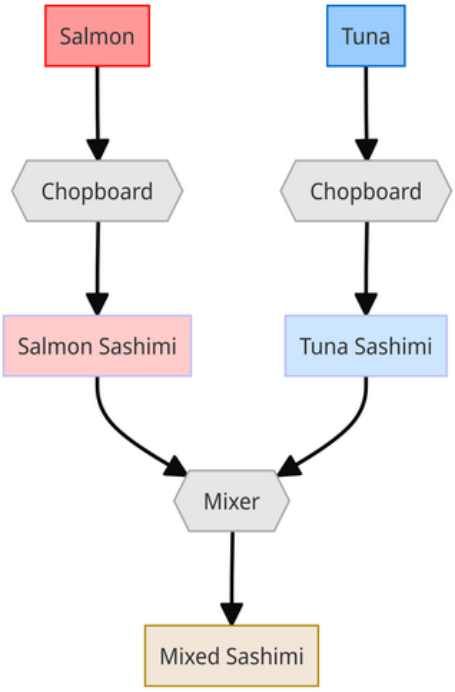
Figure B.5: Level 1 — Very Simple



(a) Salmon Sashimi

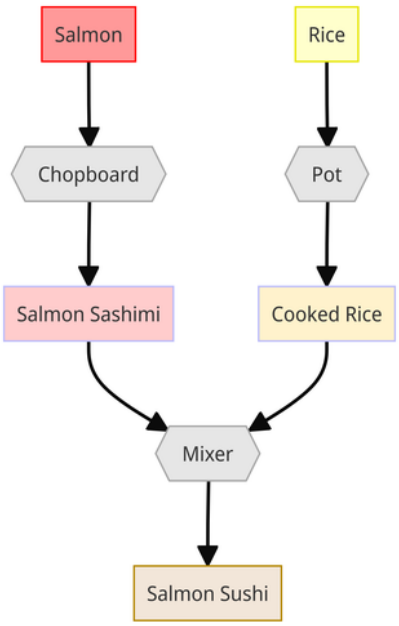


(b) Tuna Sashimi

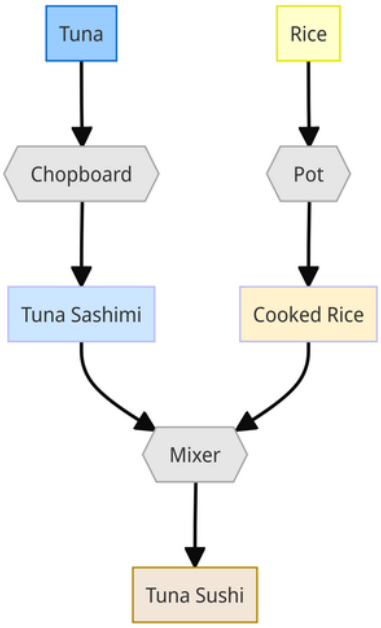


(c) Mixed Sashimi

Figure B.6: Level 2 — Simple

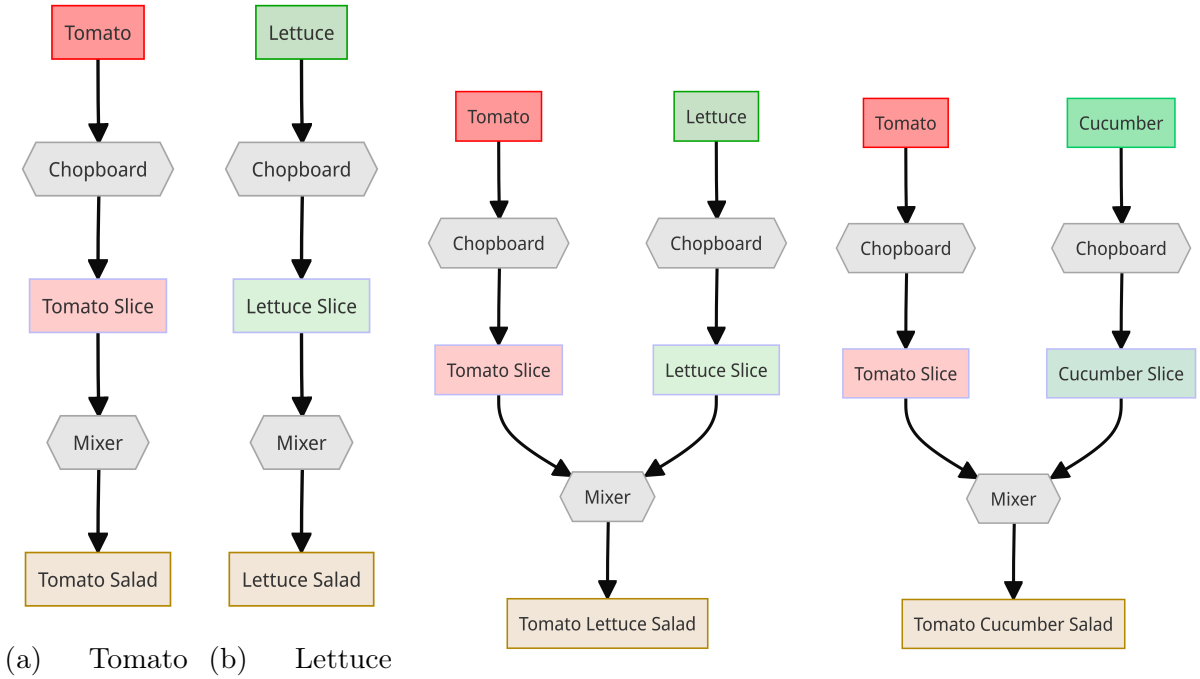


(a) Salmon Sushi



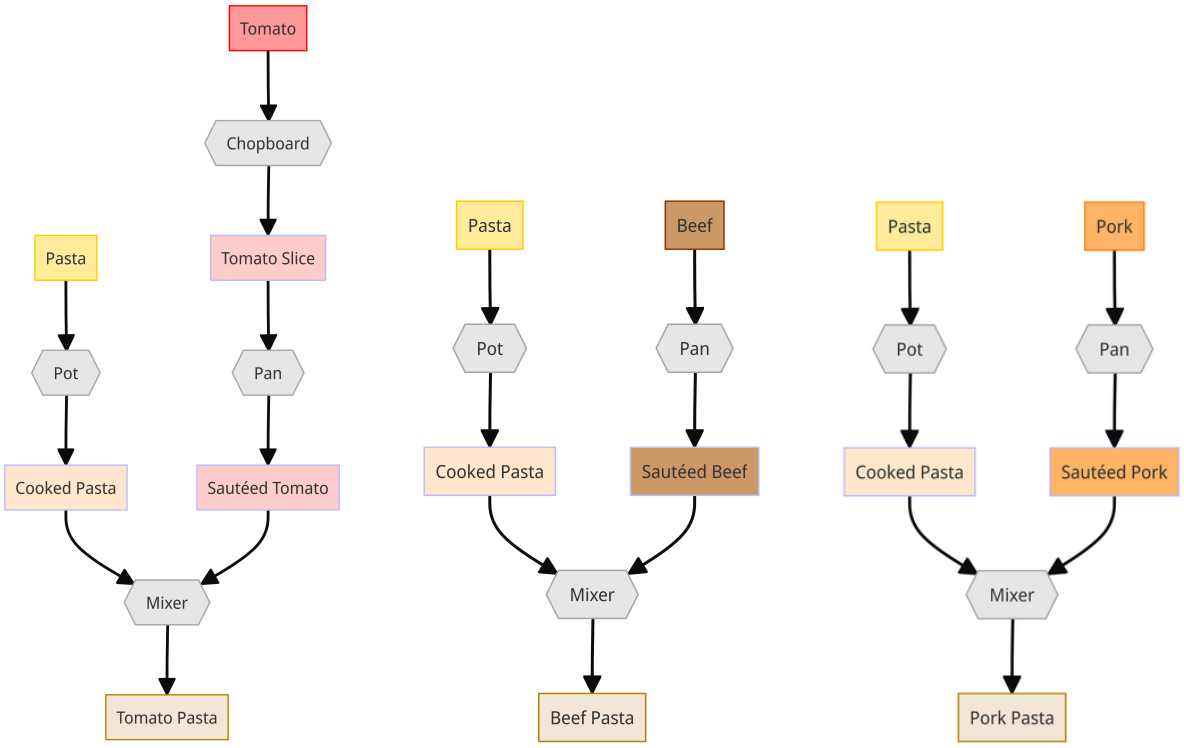
(b) Tuna Sushi

Figure B.7: Level 3 — Intermediate



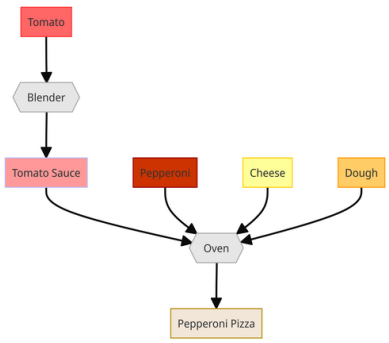
(a) Tomato Salad (b) Lettuce Salad (c) Tomato Lettuce Salad (d) Tomato Cucumber Salad

Figure B.8: Level 4 — Simple

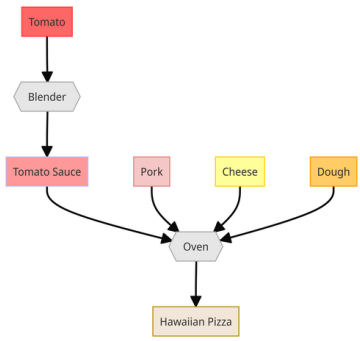


(a) Tomato Pasta (b) Beef Pasta (c) Pork Pasta

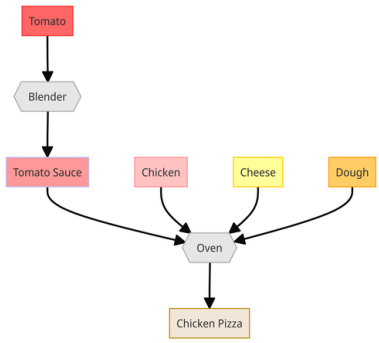
Figure B.9: Level 5 — Advanced



(a) Pepperoni Pizza

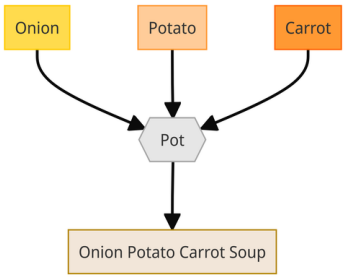


(b) Hawaiian Pizza

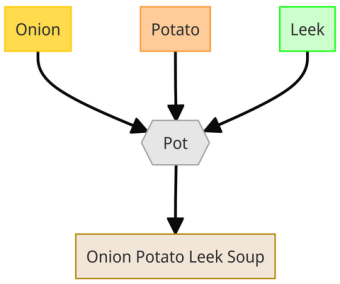


(c) Chicken Pizza

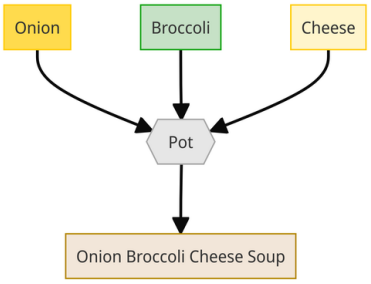
Figure B.10: Level 6 — Unused



(a) Onion Potato Carrot Soup

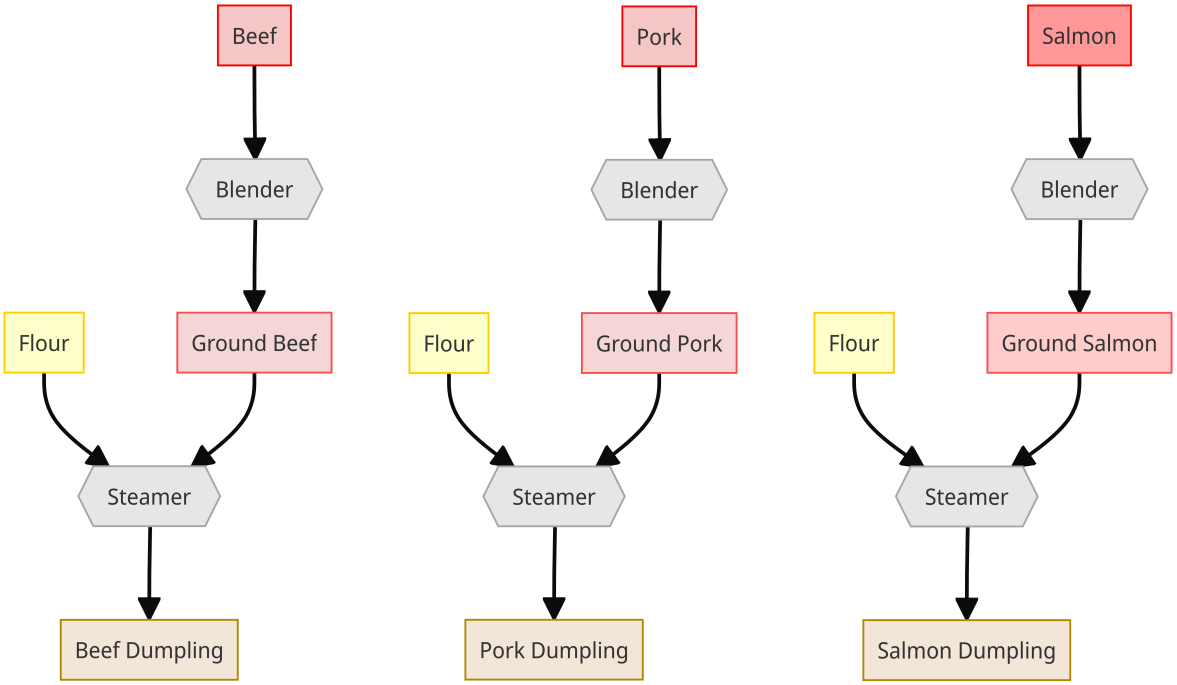


(b) Onion Potato Leek Soup



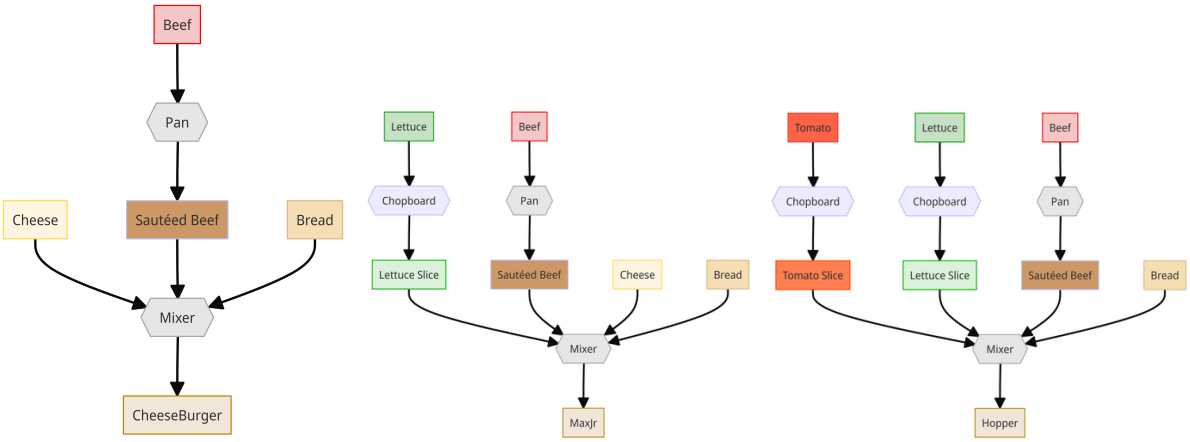
(c) Onion Broccoli Cheese Soup

Figure B.11: Level 7 — Very Simple



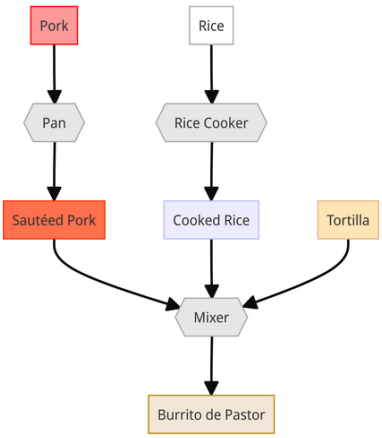
(a) Beef Dumpling (b) Pork Dumpling (c) Salmon Dumpling

Figure B.12: Level 8 — Simple

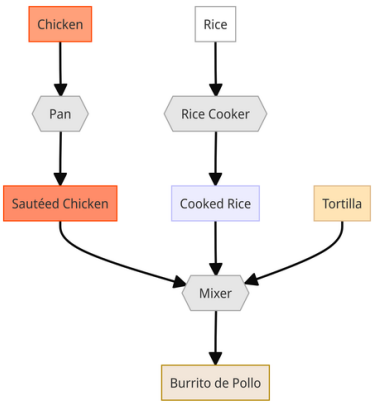


(a) Cheeseburger (b) MaxJr (c) Hopper

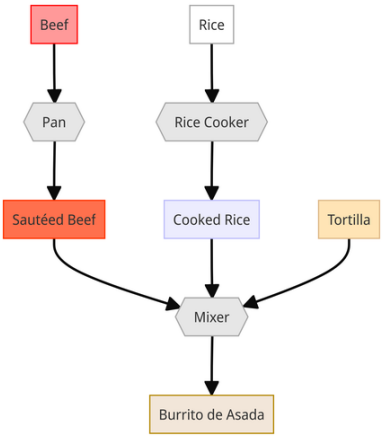
Figure B.13: Level 9 — Intermediate



(a) Burrito de Pastor

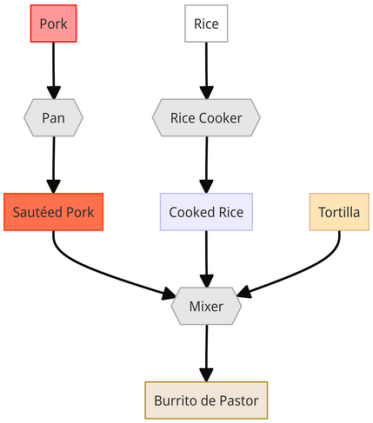


(b) Burrito de Pollo

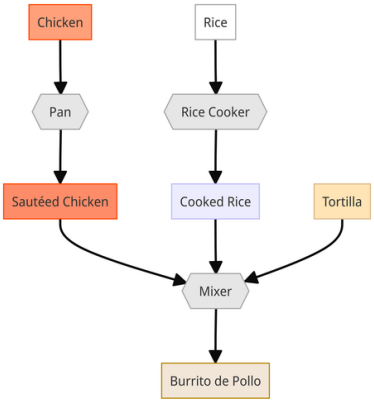


(c) Burrito de Asada

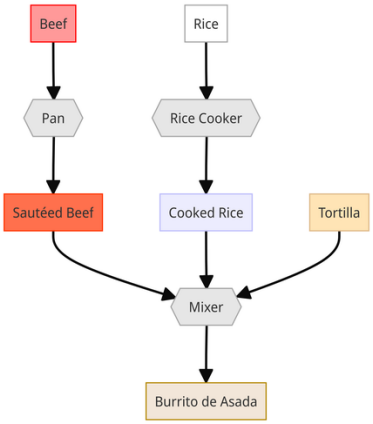
Figure B.14: Level 10 — Intermediate



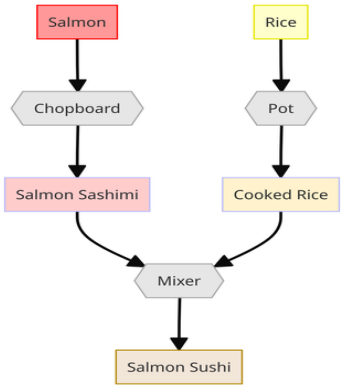
(a) Burrito de Pastor



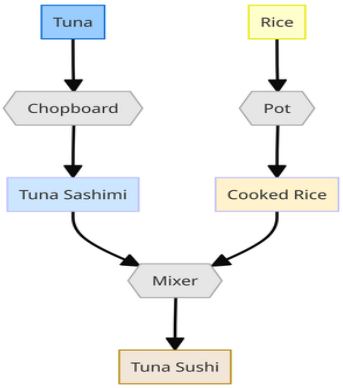
(b) Burrito de Pollo



(c) Burrito de Asada

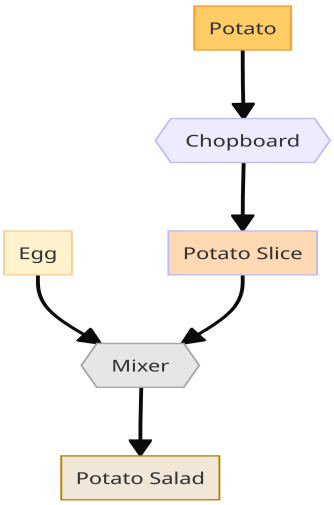


(d) Salmon Sushi

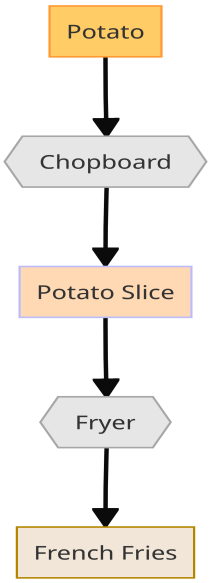


(e) Tuna Sushi

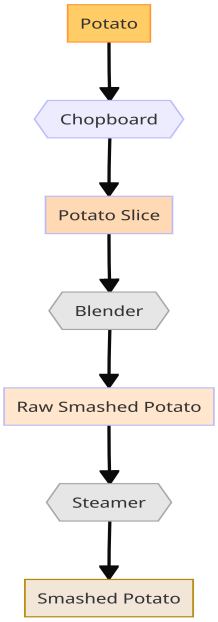
Figure B.15: Level 11 — Advanced



(a) Potato Salad



(b) French Fries



(c) Smashed Potato

Figure B.16: Level 12 — Advanced

Compared to other benchmarks, such as (Wu et al., 2021; Carroll et al., 2019), we have the following differences:

- **Enhanced Kitchen Layouts:** We have diversified the layout of kitchens by incorporating a wider range of tools (both in types and quantities) and ingredients. This approach differs from that of Wu et al. (2021) and Carroll et al. (2019), where the emphasis is not on the variety of kitchen tools and ingredients. For example, in (Carroll et al., 2019), there are only two types of ingredients and two types of tools. In (Wu et al., 2021), there are two types of ingredients, and two types of tools. In comparision, there are 27 unique ingredients, and 10 tools in CuisineWorld.

- **Complex Task Design:** Our benchmark includes a broader spectrum of recipes, varying significantly in difficulty levels. This variation is not just in terms of the number of tools and ingredients required but also in the intermediate steps involved in each recipe. Refer to Figure B.13 for a detailed illustration. This aspect of task complexity, particularly in the context of high-level planning, is not extensively explored in (Carroll et al., 2019) and (Wu et al., 2021). In (Wu et al., 2021; Carroll et al., 2019) there are very limited number of dishes, 1 and 3 respectively. However, in CuisineWorld, there are 33 unique dishes.

- **Multi-Dish Episodes and Collaborative Strategy Assessment:** We require agents to complete multiple dishes within a single episode, with the types of dishes varying to challenge and assess the collaborative strategies of the agents. Our level design ensures that there are shared intermediate steps among the types of dishes in a single episode. The system is tasked with multiple different goals at the same time. This approach allows us to use metrics like 'Collaborative Score' (CoS) to evaluate how agents collaborate to achieve higher dish throughput. This dynamic aspect of collaboration, especially in the context of dish expiration and shared tasks, offers a new dimension to the study of multi-agent cooperation, which is distinct from the environments in (Carroll et al., 2019) and (Wu et al., 2021). In (Carroll et al., 2019) the goal is to finish as many dishes as possible in a limited amount of

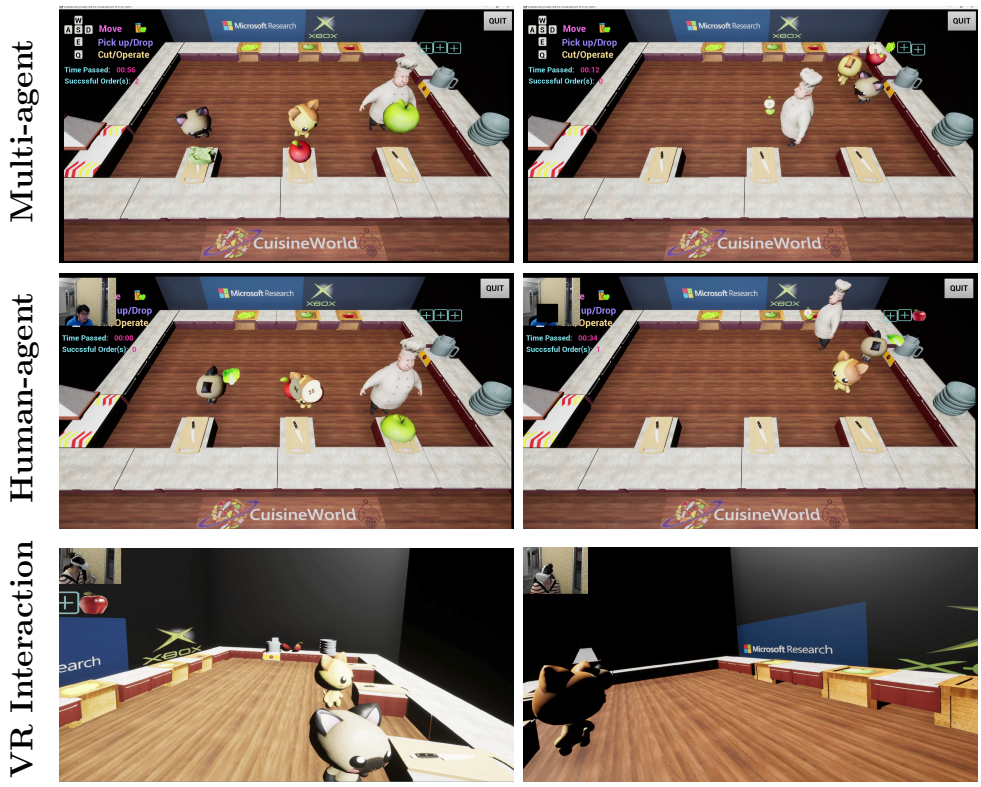


Figure B.17: (Top) A multi-agent collaboration example in `CuisineWorld`; the three agents are preparing a mixed juice together. (Middle) A human player as the head chef instructing the agents to cook mixed juice. (Bottom) A human player collaborating with collaborative agents in VR.

time. In (Wu et al., 2021), the goal is to finish one dish in the least amount of time. Both of them do not consider the density of the tasks (interval between dish orders coming to the kitchen) and its effect on coordination. As we mentioned earlier, this concpet (changing density of the tasks and measuring collaboration proficiency upon it) is at heart of CuisineWorld and our CoS metric, and it has demonstrated its effectiveness of benchmarking collaboration between LFMs and human-NPCs (as indicated in the abstract).

B.5.2 Visualizing `CuisineWorld`

To implement `CuisineWorld` into a real-world game system, we built on top of the work of Gao et al. (2020). In our game, as visually depicted in Figure B.17, players are given the

opportunity to engage in collaborative interactions with NPCs. This introduces a unique dynamic to the gameplay, enabling users to experience a more immersive cooperative environment. Additionally, the game's interface is versatile, providing players multiple ways to interact within the game world. They can either use a standard keyboard setup, which is conventional and likely familiar to most PC gamers, or immerse themselves even further using a Virtual Reality (VR) device. This VR functionality ensures a more tactile and realistic interaction, as players can physically move, gesture, and engage with the NPCs and other in-game elements in the 3D environment.

Unlike the step-by-step nature of the text version, the real-time virtual game operates continuously. To align the LFM's processing with this dynamic environment, we implemented a system where the LFM checks user actions at regular intervals during the game loop, referred to as "time steps". These time steps are defined as 0.1 seconds. Then we can ensure LFM can respond to user actions in a timely manner, matching the pace of the real-time game.

In the real-time version, human players control their agents directly through keyboard inputs, resulting in low-level actions like moving or picking up items. However, the LFM-agent operates on a higher, more temporally-extended level of atomic actions. To bridge this gap, when the LFM checks user actions, it doesn't just read the keyboard inputs. Instead, it assesses changes in the high-level game state, such as the agent's location, and the status of tools and ingredients. This assessment allows the LFM to infer the temporally extended actions that align with its text-based decision-making process. Implementing this required a simple inverse dynamics model through checking state changes to translate low-level actions into high-level atomic actions, facilitating a seamless transition from the text game to the real-time virtual game. For 'GoTo' action, we utilized the A* pathfinding algorithm, integrated into the Unreal Engine, to facilitate this movement.

B.6 Additional `CuisineWorld` Details

B.6.1 Task Interval Computation

In our approach, we initially construct a task graph delineating subgoals, which serves as the foundation for our computations. We then apply breadth-first search in a single-agent context to determine the optimal task sequence. This sequence is a key component for calculating task intervals in multi-agent collaboration scenarios. For each tool requiring activation, we incorporate its activation wait time into the respective task intervals. Additionally, for each new connection in the task graph, we increase the total task interval time by tripling the edge time. We assume each subgoal requires at least, goto, get and put 3 actions. The cumulative task interval is subsequently adjusted by a scaling factor of 0.3 and the variable τ. We pick the value *tau* ranging from 1.0, 1.5, 2.0, 2.5 and 3.0 to represent different task difficulties. This process enables us to effectively compute task intervals tailored for multi-agent collaborative environments of varied difficulties.

B.6.2 Common Failure Modes

Through replaying actions, we have identified the following common failure modes for GPT-4 agents: 1) Inability to Prioritize Task Order: Occasionally, the LFM overlooks the task at the top of the queue, leading to the expiration of that task. 2) Difficulty Understanding the 'Occupy()' State Instruction: In CuisineWorld, agents must wait for **varying** timesteps before cooking is completed, with the wait time dependent on the specific tool used. If agents attempt to remove ingredients immediately after activating a tool, the action fails. Instead of continuing to wait, the agents may shift their focus to other tasks, which slows down overall progress. 3) Challenges in Allocating Agents to Correct Subgoals: When multiple dishes are being prepared concurrently, the agents often struggle to allocate themselves effectively to the appropriate subgoals.

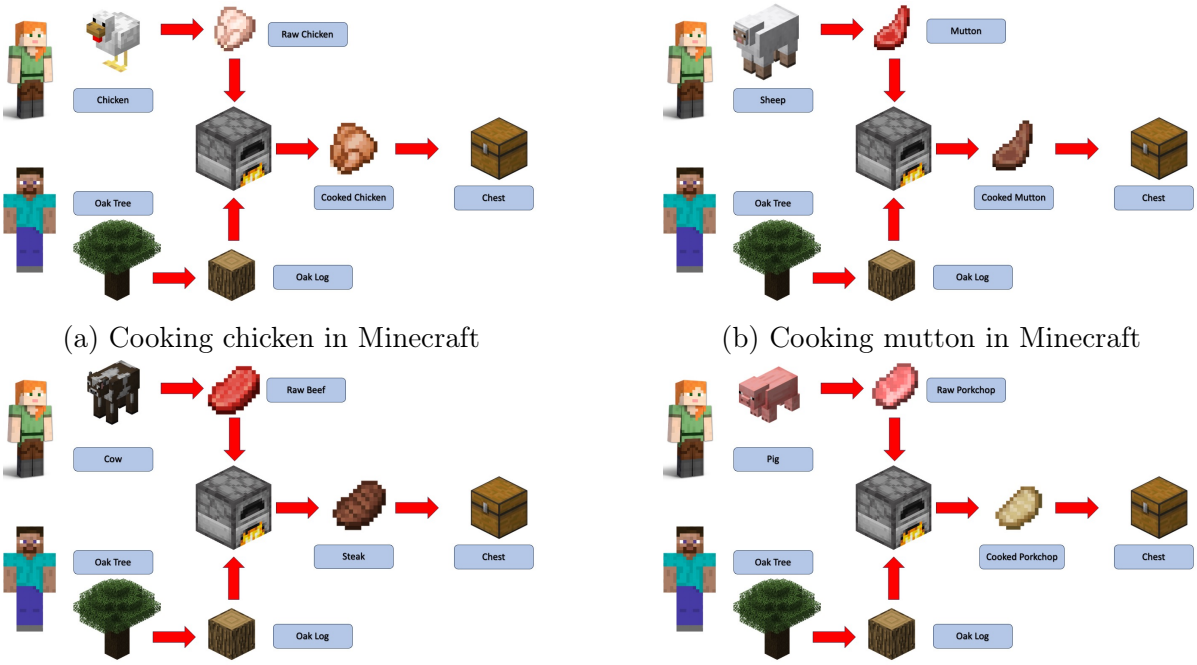
B.6.3 Rational for CoS Metric

- In the kitchen scenario as demonstrated in CuisineWorld, hypothetically, when the dish order come very rarely (with a large interval), no matter if there is any collaboration, high success rate can easily attained as there is sufficient time.
- However, as we reduce the interval, more and more dish order are flooding in. If the agents (and humans) are able to collaborate well, the productivity will be high, or namely, they can still manage to maintain a decent success rate. On the contrary, teams with poor collaborate will likely suffer from a substantial drop on success rate as the interval gets smaller. The same collapse will ultimately happen to a good collaborative team too when the interval gets too small, but good collaboration can always sustain longer.
- Therefore, it makes sense to use the averaged success rate across different intervals as an indicator of the collaboration proficiency. More importantly, such metric asks for a game setting where the dish order will keep coming, in a changing interval, which also aligns with the original Overcooked! game experiences.

B.7 Minecraft

B.7.1 Transfer to Minecraft

To transfer `MindAgent` from CuisineWorld to Minecraft requires modifications on both games and the model. On the LFM side, we update the background knowledge of the model. This included: 1) Action Space Explanation: We provided the LFM with detailed information about the possible actions and interactions within the Minecraft environment. 2) Recipe and Tool Definitions: We also included definitions of recipes, tools, and ingredients specific to Minecraft. We believe these modifications are reasonable and necessary, as without these knowledge, it's very difficult for model to operate in an unknown environments. On the game development side, we dedicated efforts to: Text-to-



(a) Cooking chicken in Minecraft

(b) Cooking mutton in Minecraft

(c) Cooking steak in Minecraft

(d) Cooking porkchop in Minecraft

Figure B.18: Task visualization in Minecraft

Game Interaction Translation: We developed code to translate text-based interactions and commands from the LFM into actionable inputs within the game environment. This translation layer was key to bridging the gap between the LFM's text-based outputs and the game's interactive elements.

B.7.2 Task Graphs

In Figure B.18 we visualize the task graphs for different tasks in Minecraft.

B.7.3 Gameplay Visualization

We visualize Minecraft gameplay in Figure B.19.

B.7.4 Action Details for Mindcraft

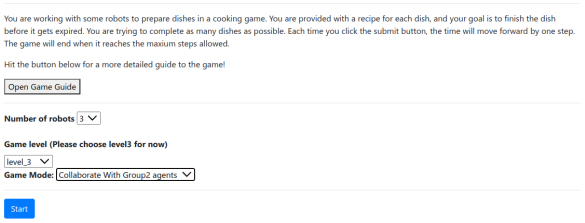
We define the following actions for the multi-agent system in our Minecraft game: 1) `goto(agent, location)`; 2) `killMob(agent, mobType)`; 3) `mineBlock(agent, blockType)`;

Figure B.19: (Top) A multi-agent collaboration example in Minecraft. At left Alex and Steve are killing different animals and at right they are cooking meat in a furnace together. (Middle) A human player instructing the agents to perform certain actions. (Bottom) A human player collaborating with agents in VR.

4) `putFuelFurnace(agent, fuelType)`, to put the item from agent's inventory to the furnace's bottom slot. 5) `putItemFurnace(agent, itemType)`, to put the item from agent's inventory to the furnace's top slot; 6) `takeOutFurnace(agent)`, take out the cooked item from the furnace 7) `putInChest(agent, itemType)`.

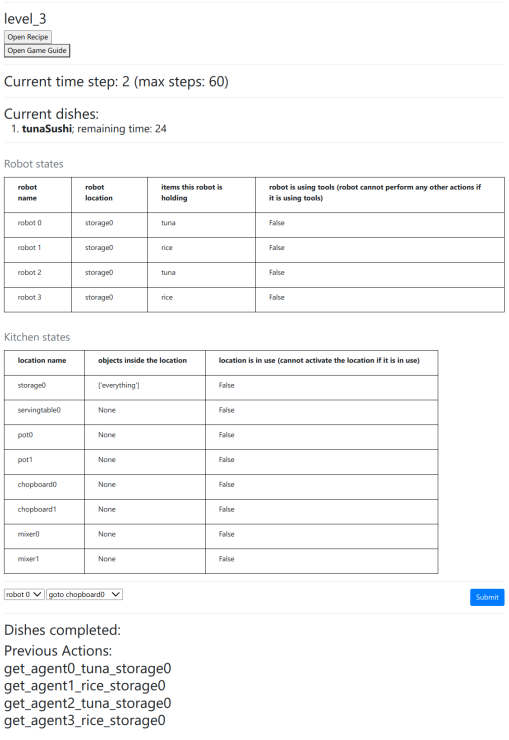
The state space in Minecraft contains the following: 1) nearby blocks for each agent, 2) nearby entities for each agent, 3) each agent's inventory, 4) items inside the furnace, 5) items inside the chest, and 6) the human player's inventory if a human player is involved.

To ensure reproducibility, we modify the game mechanism. A killed mob will respawn nearby, and a mined block will also respawn nearby.

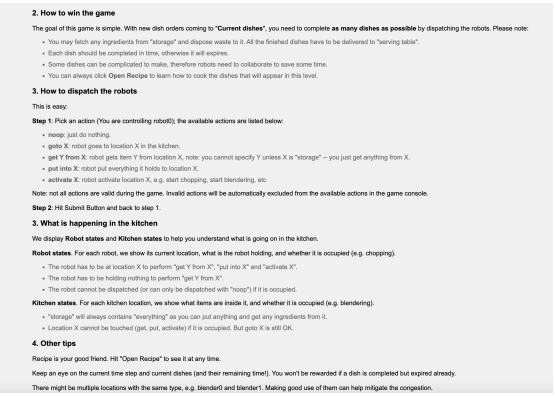


(a) Human evaluation interface welcome screen

(b) Human evaluation example



(c) Human evaluation example



(d) Human instructions

Figure B.20: Human evaluation interface welcome screen (a), evaluation examples (b)–(c), and instructions to the human participants (d).

B.8 Additional Information on Human Evaluation

B.8.1 Human Data Collection

Measurement In the background, we collect the numbers of failed and successful tasks during a participant's interaction with the game system. Additionally, we record the entire action history of players and intelligent agents. After each episode, the participants must complete a survey about their engagement with the system on a 5-point Likert chart. Our objective measure is intended to evaluate the human-AI teaming performance, and the subjective measure is designed to evaluate users' perceptions of the system. The human evaluation interface can be found in Section B.8.

B.8.2 Human Evaluation Interface

We use the human evaluation interface to test the human's perception of collaborative agents. This gives us a more controlled environment so users' perception of collaborative agents does not depend on their ability to control the keyboard and mouse, and their perception of collaborative agents does not depend on the latency and rate limits of GPT-4. Figure B.20 shows the interface welcome screen, human evaluation examples, and examples of human instructions.

B.8.3 Human Evaluation

We list our human evaluation questionnaire platform in the Figure B.21.

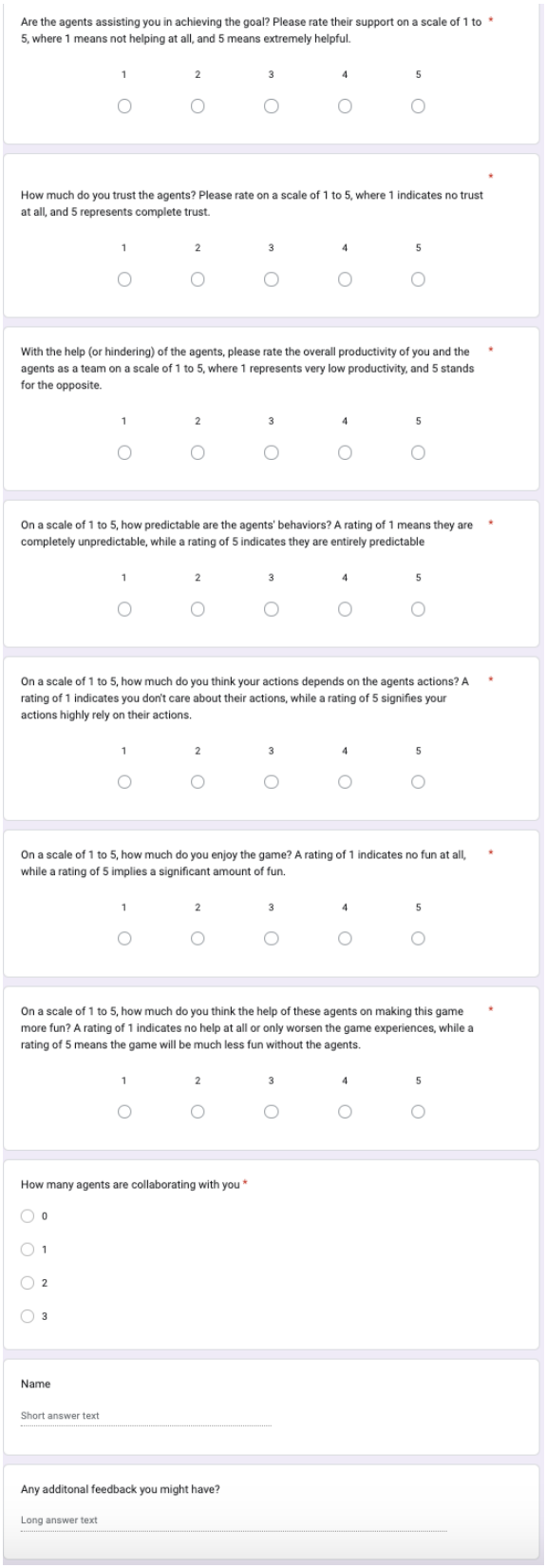


Figure B.21: Human evaluation questionnaire

Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, page 1. 10

Agrawal, P., Carreira, J., and Malik, J. (2015). Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*. 21

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., Yan, M., and Zeng, A. (2022). Do as I can and not as I say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*. 18

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and Van Den Hengel, A. (2018). Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 50, 89

Anjomshoae, S., Najjar, A., Calvaresi, D., and Främling, K. (2019). Explainable agents and robots: Results from a systematic literature review. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1078–1088. International Foundation for Autonomous Agents and Multiagent Systems. 15

Anthropic (2023). Introducing Claude. https://www.anthropic.com/index/introducing-claude. Accessed: 2023-12-15. 145

Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. (2022). Video pretraining (VPT): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654. 18

Bakhtin, A., Brown, N., Dinan, E., Farina, G., Flaherty, C., Fried, D., Goff, A., Gray, J., Hu, H., et al. (2022). Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074. 18, 151

Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., and Mordatch, I. (2017). Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*. 70

Batra, D., Chang, A. X., Chernova, S., Davison, A. J., Deng, J., Koltun, V., Levine, S., Malik, J., Mordatch, I., Mottaghi, R., et al. (2020). Rearrangement: A challenge for embodied AI. *arXiv preprint arXiv:2011.01975*. 50

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S. (2016). DeepMind lab. *arXiv preprint arXiv:1612.03801*, pages 1–11. 8, 9

Belousov, I. R., Chellali, R., and Clapworthy, G. J. (2001). Virtual reality tools for internet robotics. *Proceedings - IEEE International Conference on Robotics and Automation*. 10

Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096. 20

Brodeur, S., Perez, E., Anand, A., Golemo, F., Celotti, L., Strub, F., Rouat, J., Larochelle, H., and Courville, A. (2017). Home: A household multimodal environment. *arXiv preprint arXiv:1711.11017*. 9, 31

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., Florence, P., Fu, C., Arenas, M. G., Gopalakrishnan, K., Han, K., Hausman, K., Herzog, A., Hsu, J., Ichter, B., Irpan, A., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, L., Lee, T.-W. E., Levine, S., Lu, Y., Michalewski, H., Mordatch, I., Pertsch, K., Rao, K., Reymann, K., Ryoo, M., Salazar, G., Sanketi, P., Sermanet, P., Singh, J., Singh, A., Soricut, R., Tran, H., Vanhoucke, V., Vuong, Q., Wahid, A., Welker, S., Wohlhart, P., Wu, J., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. (2023). RT-2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint arXiv:2307.15818*. 124

Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., et al. (2022). Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning (CoRL)*. 12

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*. 106

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. (2023). Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*. 105

Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2017). Realtime multi-person 2D pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299. 47

Carreira-Perpinan, M. A. and Hinton, G. E. (2005). On contrastive divergence learning. In *Aistats*, volume 10, pages 33–40. Citeseer. 78

Carroll, M., Shah, R., Ho, M. K., Griffiths, T. L., Seshia, S. A., Abbeel, P., and Dragan, A. (2019). On the utility of learning about humans for human-AI coordination. *arXiv preprint arXiv:1910.05789*. 18, 151, 159

Chang, A., Dai, A., Funkhouser, T., Halber, M., Niebner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y. (2018). Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV 2017)*, pages 667–676. 8, 11

Chen, B., Song, S., Lipson, H., and Vondrick, C. (2020). Visual hide and seek. In *Artificial Life Conference Proceedings*, pages 645–655. MIT Press. 15

Chen, C., Schissler, C., Garg, S., Kobernik, P., Clegg, A., Calamia, P., Batra, D., Robinson, P. W., and Grauman, K. (2022a). SoundSpaces 2.0: A simulation platform for visual-acoustic learning. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track (NeurIPS Datasets and Benchmarks)*. 11

Chen, Y., Yang, Y., Wu, T., Wang, S., Feng, X., Jiang, J., McAleer, S. M., Dong, H., Lu, Z., and Zhu, S.-C. (2022b). Towards human-level bimanual dexterous manipulation with reinforcement learning. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track (NeurIPS Datasets and Benchmarks)*. 16

Chen, Z. (2012). Object-based attention: A tutorial review. *Attention, Perception, and Psychophysics*, 74(5):784–802. 40

Cheng, C.-A., Mukadam, M., Issac, J., Birchfield, S., Fox, D., Boots, B., and Ratliff, N. (2020). RMPflow: A computational graph for automatic motion policy generation. In *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*, pages 441–457. Springer. 57

Chou, W.-L. and Yeh, S.-L. (2012). Object-based attention occurs regardless of object awareness. *Psychonomic Bulletin and Review*, 19(2):225–231. 40

Côté, M.-A., Kádár, A., Yuan, X., Kybartas, B., Barnes, T., Fine, E., Moore, J., Hausknecht, M., El Asri, L., Adada, M., et al. (2019). Textworld: A learning environment for text-based games. In *Computer Games: 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised Selected Papers 7*, pages 41–75. Springer. 151

Coumans, E. and Bai, Y. (2016). Pybullet: A python module for physics simulation for games, robotics and machine learning. *GitHub repository*. 8

Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., and Batra, D. (2018). Embodied question answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 51

de Giorgio, A., Romero, M., Onori, M., and Wang, L. (2017). Human-machine collaboration in virtual reality for adaptive production engineering. *Procedia Manufacturing*. 10

Deitke, M., Batra, D., Bisk, Y., Campari, T., Chang, A. X., Chaplot, D. S., Chen, C., D'Arpino, C. P., Ehsani, K., Farhadi, A., et al. (2022a). Retrospectives on the embodied AI workshop. *arXiv preprint arXiv:2210.06849*. 50

Deitke, M., Han, W., Herrasti, A., Kembhavi, A., Kolve, E., Mottaghi, R., Salvador, J., Schwenk, D., VanderBilt, E., Wallingford, M., et al. (2020). RoboTHOR: An open simulation-to-real embodied AI platform. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 50

Deitke, M., VanderBilt, E., Herrasti, A., Weihs, L., Salvador, J., Ehsani, K., Han, W., Kolve, E., Farhadi, A., Kembhavi, A., et al. (2022b). ProcTHOR: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems (NeurIPS)*. 11

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. 1, 8, 10

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., and Su, Y. (2023). Mind2Web: Towards a generalist agent for the web. *arXiv preprint arXiv:2306.06070*. 146

Deng, Y., Guo, D., Guo, X., Zhang, N., Liu, H., and Sun, F. (2020). MQA: Answering the question via robotic manipulation. In *Robotics: Science and Systems (RSS)*. 12, 17

Dennett, D. C. (1989). *The Intentional Stance*. MIT press. 70

Devin, S. and Alami, R. (2016). An implemented theory of mind to improve human-robot shared plans execution. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 319–326. IEEE. 15

Di, Y., Zhang, R., Lou, Z., Manhardt, F., Ji, X., Navab, N., and Tombari, F. (2022). GPV-Pose: Category-level object pose estimation via geometry-guided point-wise voting. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 13

Dragan, A. and Srinivasa, S. (2013). Generating legible motion. In *Proceedings of Robotics: Science and Systems (RSS '13)*. Carnegie Mellon University. 15

Driess, D., Ha, J.-S., and Toussaint, M. (2020). Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image. In *Robotics: Science and Systems 2020 (RSS 2020)*. RSS Foundation. 16

Driess, D., Ha, J.-S., and Toussaint, M. (2021). Learning to solve sequential physical reasoning problems from a scene image. *The International Journal of Robotics Research (IJRR)*. 16

Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch,

I., and Florence, P. (2023). PaLM-E: An embodied multimodal language model. In *arXiv preprint arXiv:2303.03378*. 12

Duan, J., Yu, S., Tan, H. L., Zhu, H., and Tan, C. (2022). A survey of embodied AI: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*. 50

Ehsani, K., Han, W., Herrasti, A., VanderBilt, E., Weihs, L., Kolve, E., Kembhavi, A., and Mottaghi, R. (2021). ManipulaTHOR: A framework for visual object manipulation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 11, 51

Fan, L., Zhu, Y., Zhu, J., Liu, Z., Zeng, O., Gupta, A., Creus-Costa, J., Savarese, S., and Fei-Fei, L. (2018). Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on robot learning*, pages 767–782. PMLR. 8, 9

Fathi, A. and Rehg, J. M. (2013). Modeling actions through state changes. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 21

Feng, Y., Wu, F., Shao, X., Wang, Y., and Zhou, X. (2018). Joint 3D face reconstruction and dense alignment with position map regression network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 534–551. 46

Fu, H., Cai, B., Gao, L., Zhang, L.-X., Wang, J., Li, C., Zeng, Q., Sun, C., Jia, R., Zhao, B., et al. (2021). 3D-FRONT: 3D furnished rooms with layouts and semantics. In *International Conference on Computer Vision (ICCV)*. 52, 53

Fu, H., Xu, W., Xue, H., Yang, H., Ye, R., Huang, Y., Xue, Z., Wang, Y., and Lu, C. (2022). RFUniverse: A physics-based action-centric interactive environment for everyday household tasks. *arXiv preprint arXiv:2202.00199*. 11

Gan, C., Schwartz, J., Alter, S., Mrowca, D., Schrimpf, M., Traer, J., De Freitas, J., Kubilius, J., Bhandwaldar, A., Haber, N., et al. (2021). ThreeDWorld: A platform for interactive multi-modal physical simulation. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track (NeurIPS Datasets and Benchmarks)*. 11, 13

Gan, C., Zhou, S., Schwartz, J., Alter, S., Bhandwaldar, A., Gutfreund, D., Yamins, D. L., DiCarlo, J. J., McDermott, J., Torralba, A., et al. (2022). The ThreeDWorld transport challenge: A visually guided task-and-motion planning benchmark towards physically realistic embodied AI. In *International Conference on Robotics and Automation (ICRA)*. IEEE. 13

Gao, Q., Doering, M., Yang, S., and Chai, J. (2016). Physical causality of action verbs in grounded language understanding. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. 50

Gao, X., Gao, Q., Gong, R., Lin, K., Thattai, G., and Sukhatme, G. S. (2022). DialFRED: Dialogue-enabled agents for embodied instruction following. *IEEE Robotics and Automation Letters (RA-L)*. 50, 89, 151

Gao, X., Gong, R., Shu, T., Xie, X., Wang, S., and Zhu, S.-C. (2019). VRKitchen: An interactive 3D virtual environment for task-oriented learning. *arXiv preprint arXiv:1903.05757*. 6, 11

Gao, X., Gong, R., Zhao, Y., Wang, S., Shu, T., and Zhu, S.-C. (2020). Joint mind modeling for explanation generation in complex human-robot collaborative tasks. In *2020 29th IEEE international conference on robot and human interactive communication (RO-MAN)*, pages 1119–1126. IEEE. 7, 160

Gao, Y., Liu, F., Wang, L., Lian, Z., Wang, W., Li, S., Wang, X., Zeng, X., Wang, R., Wang, J., et al. (2023). Towards effective and interpretable human-agent collaboration in MOBA games: A communication perspective. *arXiv preprint arXiv:2304.11632*. 17

Geng, H., Li, Z., Geng, Y., Chen, J., Dong, H., and Wang, H. (2023). PartManip: Learning cross-category generalizable part manipulation policy from point cloud observations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 50

Geng, Y., An, B., Geng, H., Chen, Y., Yang, Y., and Dong, H. (2022). End-to-end affordance learning for robotic manipulation. *arXiv preprint arXiv:2209.12941*. 50

Giusti, A., Guzzi, J., Cireşan, D. C., He, F., Rodríguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Caro, G. D., Scaramuzza, D., and Gambardella, L. M. (2016). A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667. 23

Gong, R., Gao, X., Gao, Q., Shakiah, S., Thattai, G., and Sukhatme, G. S. (2023a). LEMMA: Learning language-conditioned multi-robot manipulation. *IEEE Robotics and Automation Letters*. 7

Gong, R., Huang, J., Zhao, Y., Geng, H., Gao, X., Wu, Q., Ai, W., Zhou, Z., Terzopoulos, D., Zhu, S.-C., Jia, B., and Huang, S. (2023b). ARNOLD: A benchmark for language-grounded task learning with continuous states in realistic 3D scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20483–20495. 7, 54, 56, 57, 58, 59, 61, 62, 67

Gong, R., Huang, Q., Ma, X., Vo, H., Durante, Z., Noda, Y., Zheng, Z., Zhu, S.-C., Terzopoulos, D., Fei-Fei, L., et al. (2023c). MindAgent: Emergent gaming interaction. *arXiv preprint arXiv:2309.09971*. 7

Gong, Z. and Zhang, Y. (2018). Behavior explanation as intention signaling in human-robot teaming. In *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 1005–1011. IEEE. 15

Gu, J., Chaplot, D. S., Su, H., and Malik, J. (2022). Multi-skill mobile manipulation for object rearrangement. *International Conference on Learning Representations (ICLR)*. 50

Gu, J., Xiang, F., Li, X., Ling, Z., Liu, X., Mu, T., Tang, Y., Tao, S., Wei, X., Yao, Y., et al. (2023). ManiSkill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations (ICLR)*. 12, 13, 57, 64

Gunning, D. (2017). Explainable artificial intelligence (XAI). *Defense Advanced Research Projects Agency (DARPA), 2nd Web*, 2. 70

Hadfield-Menell, D., Russell, S. J., Abbeel, P., and Dragan, A. (2016). Cooperative inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3909–3917. 14

Haidu, A., Kohlsdorf, D., and Beetz, M. (2015). Learning action failure models from interactive physics-based simulations. In *IEEE International Conference on Intelligent Robots and Systems*. 10, 21

Hawkins, K. P., Bansal, S., Vo, N. N., and Bobick, A. F. (2014). Anticipating human actions for collaboration in the presence of task and sensor uncertainty. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2215–2222. IEEE. 14

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017). Mask R-CNN. *CoRR*, abs/1703.06870. 20

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778. 47

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Sendonaris, A., Dulac-Arnold, G., Osband, I., Agapiou, J., Leibo, J. Z., and Gruslys, A. (2017). Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732. 23

Hong, Y., Wang, Z., Wu, Q., and Gould, S. (2022). Bridging the gap between learning in discrete and continuous environments for vision-and-language navigation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 51

Huang, S., Wang, Z., Li, P., Jia, B., Liu, T., Zhu, Y., Liang, W., and Zhu, S.-C. (2023). Diffusion-based generation, optimization, and planning in 3D scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 12

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. (2022a). Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9118–9147. PMLR. 18

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al. (2022b). Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning (CoRL)*. 12, 18, 89

Isola, P., Lim, J. J., and Adelson, E. H. (2015). Discovering states and transformations in image collections. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 21

Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., et al. (2022). Perceiver IO: A general architecture for structured inputs and outputs. In *International Conference on Learning Representations (ICLR)*. 60

Jain, U., Weihs, L., Kolve, E., Farhadi, A., Lazebnik, S., Kembhavi, A., and Schwing, A. (2020). A cordial sync: Going beyond marginal policies for multi-agent embodied tasks. In *European Conference on Computer Vision (ECCV)*. 15, 50, 151

Jain, U., Weihs, L., Kolve, E., Rastegari, M., Lazebnik, S., Farhadi, A., Schwing, A. G., and Kembhavi, A. (2019). Two body problem: Collaborative visual task completion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 15, 17, 50

James, S., Ma, Z., Arrojo, D. R., and Davison, A. J. (2020). RLBench: The robot learning benchmark and learning environment. *IEEE Robotics and Automation Letters (RA-L)*. 11, 13

Jang, E., Irpan, A., Khansari, M., Kappler, D., Ebert, F., Lynch, C., Levine, S., and Finn, C. (2021). BC-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning (CoRL)*. 139

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., and Fan, L. (2022). VIMA: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*. 12

Johnson, M., Hofmann, K., Hutton, T., and Bignell, D. (2016). The Malmo platform for artificial intelligence experimentation. *IJCAI International Joint Conference on Artificial Intelligence*, 2016-Janua:4246–4247. 8, 9

Kamath, A., Singh, M., LeCun, Y., Synnaeve, G., Misra, I., and Carion, N. (2021). MDETR-modulated detection for end-to-end multi-modal understanding. In *International Conference on Computer Vision (ICCV)*. 50

Kawasaki, H., Nakayama, K., Mouri, T., and Ito, S. (2001). Virtual teaching based on hand manipulability for multi-fingered robots. *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*. 10

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaskowski, W. (2017). ViZDoom: A doom-based ai research platform for visual reinforcement learning. *IEEE Conference on Computatonal Intelligence and Games, CIG*. 8, 9

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., and Farhadi, A. (2017). AI2-THOR: An interactive 3D environment for visual AI. *arXiv preprint arXiv:1712.05474*. 9, 11, 31, 33, 52, 54

Koppula, H. S., Gupta, R., and Saxena, A. (2013). Learning human activities and object affordances from RGB-D videos. *The International Journal of Robotics Research*, 32(8):951–970. 10

Korsah, G. A., Stentz, A., and Dias, M. B. (2013). A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512. 92

Krantz, J., Wijmans, E., Majumdar, A., Batra, D., and Lee, S. (2020). Beyond the Nav-Graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision (ECCV)*. 51

Krizhevsky, A. and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Neural Information Processing Systems*. 20

Kumar, V. and Todorov, E. (2015). MuJoCo HAPTIX: A virtual reality system for hand manipulation. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. 51

Kwon, M., Huang, S. H., and Dragan, A. D. (2018). Expressing robot incapability. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 87–95. 15

Langley, P., Meadows, B., Sridharan, M., and Choi, D. (2017). Explainable agency for intelligent autonomous systems. In *Twenty-Ninth IAAI Conference*. 15

Lerer, A., Gross, S., and Fergus, R. (2016). Learning physical intuition of block towers by example. In *International Conference on Machine Learning*, pages 430–438. PMLR. 10, 21

Lertkultanon, P. and Pham, Q.-C. (2018). A certified-complete bimanual manipulation planner. *IEEE Transactions on Automation Science and Engineering*, 15(3):1355–1368. 16

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373. 70

Li, C., Xia, F., Martín-Martín, R., Lingelbach, M., Srivastava, S., Shen, B., Vainio, K., Gokmen, C., Dharan, G., Jain, T., et al. (2021). iGibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *Conference on Robot Learning (CoRL)*. 11, 50

Li, C., Zhang, R., Wong, J., Gokmen, C., Srivastava, S., Martín-Martín, R., Wang, C., Levine, G., Lingelbach, M., Sun, J., et al. (2022). BEHAVIOR-1K: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning (CoRL)*. 11, 13

Li, G., Hammoud, H. A. A. K., Itani, H., Khizbullin, D., and Ghanem, B. (2023). Camel: Communicative agents for "mind" exploration of large scale language model society. *arXiv preprint arXiv:2303.17760*. 18

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. (2022). Code as policies: Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*. 18, 105

Lin, J., Guo, X., Shao, J., Jiang, C., Zhu, Y., and Zhu, S.-C. (2016). A virtual reality platform for dynamic human-scene interaction. In *SIGGRAPH ASIA 2016 Virtual Reality Meets Physical Reality: Modelling and Simulating Virtual Humans and Environments*, page 11. ACM. 8

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988. 46, 47

Lin, X., Wang, Y., Olkin, J., and Held, D. (2020). SoftGym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning (CoRL)*. 12, 13, 51

Liu, C., Hamrick, J. B., Fisac, J. F., Dragan, A. D., Hedrick, J. K., Sastry, S. S., and Griffiths, T. L. (2016). Goal inference improves objective and perceived performance in human-robot collaboration. In *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems*, pages 940–948. International Foundation for Autonomous Agents and Multiagent Systems. 14

Liu, H., Zhang, Y., Si, W., Xie, X., Zhu, Y., and Zhu, S.-C. (2018). Interactive robot knowledge patching using augmented reality. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1947–1954. IEEE. 72

Liu, M., Li, X., Ling, Z., Li, Y., and Su, H. (2022a). Frame mining: A free lunch for learning robotic manipulation from 3D point clouds. In *Conference on Robot Learning (CoRL)*. 54

Liu, O., Rakita, D., Mutlu, B., and Gleicher, M. (2017a). Understanding human-robot interaction in virtual reality. In *RO-MAN 2017 - 26th IEEE International Symposium on Robot and Human Interactive Communication*. 10

Liu, X., Guo, D., Liu, H., and Sun, F. (2022b). Multi-agent embodied visual semantic navigation with scene prior knowledge. *IEEE Robotics and Automation Letters*, 7(2):3154–3161. 15

Liu, X., Li, X., Guo, D., Tan, S., Liu, H., and Sun, F. (2022c). Embodied multi-agent task planning from ambiguous instruction. *Proceedings of Robotics: Science and Systems, New York City, NY, USA*, pages 1–14. 12, 15, 17, 151

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., et al. (2023). AgentBench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*. 146

Liu, Y., Wei, P., and Zhu, S.-C. (2017b). Jointly recognizing object fluents and tasks in egocentric videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2924–2932. 13, 21

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169. 53

Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in Neural Information Processing Systems*, 30. 17

Lu, J., Batra, D., Parikh, D., and Lee, S. (2019). ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in Neural Information Processing Systems (NeurIPS)*. 50

Lynch, C. and Sermanet, P. (2021). Language conditioned imitation learning over unstructured data. *Robotics: Science and Systems (RSS)*. 12

Lynch, C., Wahid, A., Tompson, J., Ding, T., Betker, J., Baruch, R., Armstrong, T., and Florence, P. (2022). Interactive language: Talking to robots in real time. *arXiv preprint arXiv:2210.06407*. 12

Ma, L., Meng, J., Liu, S., Chen, W., Xu, J., and Chen, R. (2023). Sim2Real^2: Actively building explicit physics model for precise articulated object manipulation. In *International Conference on Robotics and Automation (ICRA)*. 13

Ma, X., Yong, S., Zheng, Z., Li, Q., Liang, Y., Zhu, S.-C., and Huang, S. (2022). SQA3D: Situated question answering in 3D scenes. In *International Conference on Learning Representations (ICLR)*. 51

Macklin, M. and Müller, M. (2013). Position based fluids. *ACM Transactions on Graphics (TOG)*, 32(4):1–12. 53

Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., et al. (2021). Isaac Gym: High performance GPU-based physics simulation for robot learning. *Advances in Neural Information Processing Systems Datasets and Benchmarks Track (NeurIPS Datasets and Benchmarks)*. 53

McCormac, J., Handa, A., Leutenegger, S., and Davison, A. J. (2017). SceneNet RGB-D: Can 5M synthetic images beat generic imagenet pre-training on indoor segmentation? In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 2697–2706. 9

Mees, O., Hermann, L., and Burgard, W. (2022a). What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters (RA-L)*. 51

Mees, O., Hermann, L., Rosete-Beas, E., and Burgard, W. (2022b). CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*. 12, 13, 17, 51, 89, 93

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38. 15

Min, S. Y., Chaplot, D. S., Ravikumar, P., Bisk, Y., and Salakhutdinov, R. (2022). Film: Following instructions in language with modular methods. In *International Conference on Learning Representations (ICLR)*. 93

Mirchandani, S., Xia, F., Florence, P., Ichter, B., Driess, D., Arenas, M. G., Rao, K., Sadigh, D., and Zeng, A. (2023). Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*. 105

Mittal, M., Yu, C., Yu, Q., Liu, J., Rudin, N., Hoeller, D., Yuan, J. L., Tehrani, P. P., Singh, R., Guo, Y., et al. (2023). ORBIT: A unified simulation framework for interactive robot learning environments. *arXiv preprint arXiv:2301.04195*. 13

Mu, T., Ling, Z., Xiang, F., Yang, D., Li, X., Tao, S., Huang, Z., Jia, Z., and Su, H. (2021). ManiSkill: Generalizable manipulation skill benchmark with large-scale demonstrations. *Advances in Neural Information Processing Systems Datasets and Benchmarks Track (NeurIPS Datasets and Benchmarks)*. 13, 53, 57, 64

Nagarajan, T. and Grauman, K. (2018). Attributes as operators: Factorizing unseen attribute-object compositions. In *European Conference on Computer Vision (ECCV)*. 13

Nagarajan, T. and Grauman, K. (2020). Learning affordance landscapes for interaction exploration in 3D environments. In *Advances in Neural Information Processing Systems (NeurIPS)*. 50

Nair, S., Mitchell, E., Chen, K., Savarese, S., Finn, C., et al. (2022). Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning (CoRL)*. 12

Nan, Z., Shu, T., Gong, R., Wang, S., Wei, P., Zhu, S.-C., and Zheng, N. (2020). Learning to infer human attention in daily activities. *Pattern Recognition*, 103:107314. 6

Ng, A. Y. and Russell, S. (2000). Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*. 23

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. (2011). Multimodal deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 689–696. 20

Nikolaidis, S., Kwon, M., Forlizzi, J., and Srinivasa, S. (2018). Planning with verbal communication for human-robot collaboration. *ACM Transactions on Human-Robot Interaction (THRI)*, 7(3):1–21. 15

OpenAI (2023). GPT-4 technical report. 145

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Gray, A., et al. (2022). Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*. 145

Padmakumar, A., Thomason, J., Shrivastava, A., Lange, P., Narayan-Chen, A., Gella, S., Piramuthu, R., Tur, G., and Hakkani-Tur, D. (2022). TEACh: Task-driven embodied agents that chat. In *AAAI Conference on Artificial Intelligence (AAAI)*. 50, 151

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*. 17, 151

Pashevich, A., Schmid, C., and Sun, C. (2021). Episodic transformer for vision-and-language navigation. In *International Conference on Computer Vision (ICCV)*. 50, 99

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. (2018). Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*. 8

Pooresmaeili, A. and Roelfsema, P. R. (2014). A growth-cone model for the spread of object-based attention during contour grouping. *Current Biology*, 24(24):2869–2877. 40

Premack, D. and Woodruff, G. (1978). Does the chimpanzee have a theory of mind? *Behavioral and Brain Sciences*, 1(4):515–526. 70

Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., and Torralba, A. (2018). VirtualHome: Simulating household activities via programs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 9, 11

Puig, X., Shu, T., Li, S., Wang, Z., Tenenbaum, J. B., Fidler, S., and Torralba, A. (2020). Watch-And-Help: A challenge for social perception and human-AI collaboration. *arXiv preprint arXiv:2010.09890*. 18, 151

Qiu, W. and Yuille, A. (2016). UnrealCV: Connecting computer vision to unreal engine. In *Lecture Notes in Computer Science*, volume 9915, pages 909–916. Springer. 10

Qureshi, F. and Terzopoulos, D. (2008). Smart camera networks in virtual reality. *Proceedings of the IEEE*, 96(10):1640–1656. 2, 8

Rabie, T. F. and Terzopoulos, D. (2000). Active perception in virtual humans. In *Vision Interface*, volume 2000. 2, 8

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*. 50, 60, 66, 139

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*. 66, 141

Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. (2020). Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1):7234–7284. 17

Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., and Ré, C. (2017). Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282. 11

Reddy, S., Dragan, A., and Levine, S. (2018). Where do you think you're going?: Inferring beliefs about dynamics from behavior. In *Advances in Neural Information Processing Systems*, pages 1454–1465. 14

Rohrbach, M., Amin, S., Andriluka, M., and Schiele, B. (2012). A database for fine grained activity detection of cooking activities. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1194–1201. 8

Ross, S., Gordon, G. J., and Bagnell, J. A. (2010). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of AISTATS*, volume 15, pages 627–635. 10, 36

Ruiz, N., Chong, E., and Rehg, J. M. (2018). Fine-grained head pose estimation without keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2074–2083. 47

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., et al. (2022). Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*. 50

Savva, M., Chang, A. X., Dosovitskiy, A., Funkhouser, T., and Koltun, V. (2017). MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*. 9

Seemann, A. (2011). *Joint attention: New Developments in Psychology, Philosophy of Mind, and Social Neuroscience*. MIT Press. 40

Sharma, V., Goyal, P., Lin, K., Thattai, G., Gao, Q., and Sukhatme, G. S. (2022). CH-MARL: A multimodal benchmark for cooperative, heterogeneous multi-agent reinforcement learning. *arXiv preprint arXiv:2208.13626*. 17

Shirley, P. and Morley, R. K. (2008). *Realistic Ray Tracing*. AK Peters, Ltd. 54

Shridhar, M., Manuelli, L., and Fox, D. (2022a). CLIPort: What and where pathways for robotic manipulation. In *Conference on Robot Learning (CoRL)*. 12, 13, 50, 51, 89, 95, 98, 100, 140

Shridhar, M., Manuelli, L., and Fox, D. (2022b). Perceiver-Actor: A multi-task transformer for robotic manipulation. *Conference on Robot Learning (CoRL)*. 12, 50, 51, 54, 60, 61, 62, 128, 139

Shridhar, M., Manuelli, L., and Fox, D. (2023). Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR. 89

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. (2020a). ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 12, 13, 17, 51, 89

Shridhar, M., Yuan, X., Côté, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. (2020b). ALFworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*. 151

Shu, T., Gao, X., Ryoo, M. S., and Zhu, S.-C. (2017). Learning social affordance grammar from videos: Transferring human interactions to human-robot interactions. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1669–1676. IEEE. 72

Shu, T., Xie, D., Rothrock, B., Todorovic, S., and Zhu, S.-C. (2015). Joint inference of groups, events and human roles in aerial videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4576–4584. 72

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359. 20

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. (2023). Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE. 16

Smith, C., Karayiannidis, Y., Nalpantidis, L., Gratal, X., Qi, P., Dimarogonas, D. V., and Kragic, D. (2012). Dual arm manipulation—a survey. *Robotics and Autonomous systems*, 60(10):1340–1353. 16

Smith, L. and Gasser, M. (2005). The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29. 1, 2, 20, 21, 122

Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., and Funkhouser, T. (2017). Semantic scene completion from a single depth image. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 190–198. 9, 28

Sreedharan, S., Srivastava, S., and Kambhampati, S. (2018). Hierarchical expertise level modeling for user specific contrastive explanations. In *IJCAI*, pages 4829–4836. 15

Srivastava, S., Li, C., Lingelbach, M., Martín-Martín, R., Xia, F., Vainio, K. E., Lian, Z., Gokmen, C., Buch, S., Liu, K., et al. (2022). BEHAVIOR: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on Robot Learning (CoRL)*. 11, 13, 51

Stavridis, S., Falco, P., and Doulgeri, Z. (2021). Pick-and-place in dynamic environments with a mobile dual-arm robot equipped with distributed distance sensors. In *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pages 76–82. IEEE. 16

Stepputtis, S., Bandari, M., Schaal, S., and Amor, H. B. (2022). A system for imitation learning of contact-rich bimanual manipulation policies. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11810–11817. IEEE. 16

Stepputtis, S., Campbell, J., Phielipp, M., Lee, S., Baral, C., and Ben Amor, H. (2020). Language-conditioned imitation learning for robot manipulation tasks. *Advances in Neural Information Processing Systems (NeurIPS)*. 12

Stone, P., Kaminka, G. A., Kraus, S., and Rosenschein, J. S. (2010). Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*. 15

Stone, P. and Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8:345–383. 105

Struckmeier, O., Racca, M., and Kyrki, V. (2019). Autonomous generation of robust and focused explanations for robot policies. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1–8. IEEE. 15

Suhr, A., Yan, C., Schluger, C., Yu, S., Khader, H., Mouallem, M., Zhang, I., and Artzi, Y. (2019). Executing instructions in situated collaborative interactions. *arXiv preprint arXiv:1910.03655*. 151

Syed, U. and Schapire, R. E. (2008). A game-theoretic approach to apprenticeship learning. In *Advances in Neural Information Processing Systems 20*, volume 20, pages 1–8. 10

Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J., Maestre, N., Mukadam, M., Chaplot, D. S., Maksymets, O., et al. (2021). Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems (NeurIPS)*. 11, 12, 13, 15, 51

Tabrez, A., Agrawal, S., and Hayes, B. (2019). Explanation-based reward coaching to improve human performance via reinforcement learning. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 249–257. IEEE. 15

Takata, K., Kiyokawa, T., Ramirez-Alpizar, I. G., Yamanobe, N., Wan, W., and Harada, K. (2022). Efficient task/motion planning for a dual-arm robot from language instructions and cooking images. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12058–12065. IEEE. 16

Tan, S., Xiang, W., Liu, H., Guo, D., and Sun, F. (2020). Multi-agent embodied question answering in interactive environments. In *European Conference on Computer Vision*, pages 663–678. Springer. 12, 15, 17

Terzopoulos, D. and Rabie, T. F. (1995). Animat vision: Active vision in artificial animals. In *Proceedings of IEEE International Conference on Computer Vision*, pages 801–808. IEEE. 2, 8

Thomason, J., Murray, M., Cakmak, M., and Zettlemoyer, L. (2020). Vision-and-dialog navigation. In *Conference on Robot Learning (CoRL)*. 50

Todorov, E., Erez, T., and Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. In *IEEE International Conference on Intelligent Robots and Systems*. 8

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*. 145

Tseng, W.-C., Liao, H.-J., Yen-Chen, L., and Sun, M. (2022). CLA-NeRF: Category-level articulated neural radiance field. In *International Conference on Robotics and Automation (ICRA)*. 13

Tu, K., Pavlovskaia, M., and Zhu, S.-C. (2013). Unsupervised structure learning of stochastic and-or grammars. In *Advances in neural information processing systems*, pages 1322–1330. 72

Urbanek, J., Fan, A., Karamcheti, S., Jain, S., Humeau, S., Dinan, E., Rocktäschel, T., Kiela, D., Szlam, A., and Weston, J. (2019). Learning to speak and act in a fantasy text adventure game. *arXiv preprint arXiv:1903.03094*. 151

von Ahn, L. and Dabbish, L. (2008). Designing games with a purpose. *Communications of the ACM*. 23

Wan, W., Geng, H., Liu, Y., Shan, Z., Yang, Y., Yi, L., and Wang, H. (2023). UniDexGrasp++ : Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. *arXiv preprint arXiv:2304.00464*. 50

Wan, Y., Mao, J., and Tenenbaum, J. (2022). HandMeThat: Human-robot communication in physical and social environments. *Advances in Neural Information Processing Systems*, 35:12014–12026. 18, 151

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. (2023a). Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*. 18, 105, 145

Wang, H., Wang, W., Zhu, X., Dai, J., and Wang, L. (2021). Collaborative visual navigation. *arXiv preprint arXiv:2107.01151*. 15

Wang, N., Pynadath, D. V., and Hill, S. G. (2016). Trust calibration within a human-robot team: Comparing automatically generated explanations. In *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*, pages 109–116. IEEE Press. 15

Wang, S., Han, M., Jiao, Z., Zhang, Z., Wu, Y. N., Zhu, S.-C., and Liu, H. (2024). LLM 3: Large language model-based task and motion planning with motion failure reasoning. *arXiv preprint arXiv:2403.11552*. 18

Wang, X., Huang, Q., Celikyilmaz, A., Gao, J., Shen, D., Wang, Y.-F., Wang, W. Y., and Zhang, L. (2019). Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 50

Wang, Z., Cai, S., Liu, A., Ma, X., and Liang, Y. (2023b). Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*. 18, 105

Wang, Z., Chen, Y., Liu, T., Zhu, Y., Liang, W., and Huang, S. (2022). HUMANISE: Language-conditioned human motion generation in 3D scenes. In *Advances in Neural Information Processing Systems (NeurIPS)*. 50

Wei, F., Chabra, R., Ma, L., Lassner, C., Zollhoefer, M., Rusinkiewicz, S., Sweeney, C., Newcombe, R., and Slavcheva, M. (2022a). Self-supervised neural articulated shape and appearance models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 13

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2021). Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*. 106

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022b). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837. 106

Weng, Y., Wang, H., Zhou, Q., Qin, Y., Duan, Y., Fan, Q., Chen, B., Su, H., and Guibas, L. J. (2021). CAPTRA: Category-level pose tracking for rigid and articulated objects from point clouds. In *International Conference on Computer Vision (ICCV)*. 13

Wu, S. A., Wang, R. E., Evans, J. A., Tenenbaum, J. B., Parkes, D. C., and Kleiman-Weiner, M. (2021). Too many cooks: Bayesian inference for coordinating multi-agent collaboration. *Topics in Cognitive Science*, 13(2):414–432. 17, 151, 159, 160

Wu, Y., Wu, Y., Gkioxari, G., and Tian, Y. (2018). Building generalizable agents with a realistic and rich 3D environment. *arXiv preprint arXiv:1801.02209*. 9

Xia, F., Shen, W. B., Li, C., Kasimbeg, P., Tchapmi, M. E., Toshev, A., Martín-Martín, R., and Savarese, S. (2020). Interactive Gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters (RA-L)*. 50

Xia, F., Zamir, A. R., He, Z., Sax, A., Malik, J., and Savarese, S. (2018). Gibson Env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9

Xiang, F., Qin, Y., Mo, K., Xia, Y., Zhu, H., Liu, F., Liu, M., Jiang, H., Yuan, Y., Wang, H., et al. (2020). SAPIEN: A simulated part-based interactive environment. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 52, 54

Xing, E., Gupta, A., Powers, S., and Dean, V. (2021). KitchenShift: Evaluating zero-shot generalization of imitation-based policy learning under domain shifts. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*. 13

Xiong, C., Shukla, N., Xiong, W., and Zhu, S.-C. (2016). Robot learning with a spatial, temporal, and causal and-or graph. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2144–2151. IEEE. 72

Xu, A. and Dudek, G. (2015). Optimo: Online probabilistic trust inference model for asymmetric human-robot collaborations. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 221–228. IEEE. 15

Xu, Y., Wan, W., Zhang, J., Liu, H., Shan, Z., Shen, H., Wang, R., Geng, H., Weng, Y., Chen, J., et al. (2023). UniDexGrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. *arXiv preprint arXiv:2303.00938*. 50

Yao, B., Yang, X., and Zhu, S.-C. (2007). Introduction to a large-scale general purpose ground truth database: Methodology, annotation tool and benchmarks. In *Energy Minimization Methods in Computer Vision and Pattern Recognition: 6th International Conference, EMMCVPR 2007, Ezhou, China, August 27-29, 2007. Proceedings 6*, pages 169–183. Springer. 2

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*. 18, 105, 146

Yuan, L., Gao, X., Zheng, Z., Edmonds, M., Wu, Y. N., Rossano, F., Lu, H., Zhu, Y., and Zhu, S.-C. (2022). In situ bidirectional human-robot value alignment. *Science robotics*, 7(68):eabm4183. 117

Zeng, A., Attarian, M., Choromanski, K. M., Wong, A., Welker, S., Tombari, F., Purohit, A., Ryoo, M. S., Sindhwani, V., Lee, J., et al. (2022). Socratic models: Composing zero-shot multimodal reasoning with language. In *The Eleventh International Conference on Learning Representations*. 12

Zeng, A., Florence, P., Tompson, J., Welker, S., Chien, J., Attarian, M., Armstrong, T., Krasin, I., Duong, D., Sindhwani, V., et al. (2021). Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning (CoRL)*. 13, 17, 60, 93

Zhang, C., Yang, K., Hu, S., Wang, Z., Li, G., Sun, Y., Zhang, C., Zhang, Z., Liu, A., Zhu, S.-C., et al. (2023a). Proagent: Building proactive cooperative AI with large language models. *arXiv preprint arXiv:2308.11339*. 17

Zhang, H., Du, W., Shan, J., Zhou, Q., Du, Y., Tenenbaum, J. B., Shu, T., and Gan, C. (2023b). Building cooperative embodied agents modularly with large language models. *arXiv preprint arXiv:2307.02485*. 17, 151

Zhang, H., Lai, P.-J., Paul, S., Kothawade, S., and Nikolaidis, S. (2019). Learning collaborative action plans from YouTube videos. In *The International Symposium of Robotics Research*, pages 208–223. Springer. 16

Zhang, Y. and Chai, J. (2021). Hierarchical task learning from language instructions with unified transformers and self-monitoring. In *Findings of the Association for Computational Linguistics (ACL-IJCNLP Findings)*. 50

Zhang, Y., Sreedharan, S., Kulkarni, A., Chakraborti, T., Zhuo, H. H., and Kambhampati, S. (2017). Plan explicability and predictability for robot task planning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1313–1320. IEEE. 15

Zheng, K., Chen, X., Jenkins, O. C., and Wang, X. E. (2022). VLMbench: A compositional benchmark for vision-and-language manipulation. *Advances in Neural Information Processing Systems Datasets and Benchmarks Track (NeurIPS Datasets and Benchmarks)*. 12, 13, 17, 50, 51, 54, 60, 62, 89, 93, 128

Zhu, Y., Gordon, D., Kolve, E., Fox, D., Fei-Fei, L., Gupta, A., Mottaghi, R., and Farhadi, A. (2017). Target-driven visual navigation in indoor scenes using deep reinforcement learning. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob(1):483–492. 23

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, pages 1433–1438. 23